RL-TR-96-242
Final Technical Report
February 1997

SPACE EFFECTIVENESS ANALYSIS SIMULATION (SEAS) SYSTEM

Synectics Corporation

Dr. A.E.R. Woodcock and Dr. Loren Cobb

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-96-242 has been reviewed and is approved for publication.

APPROVED:

PETER J. COSTIANES
Project Engineer

FOR THE COMMANDER:

JOSEPH CAMERA, Technical Director
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/IRRE, 32 Hangar Road, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1997	Final Sep 94 - Jul 96

4. TITLE AND SUBTITLE
SPACE EFFECTIVENESS ANALYSIS SIMULATION (SEAS) SYSTEM

5. FUNDING NUMBERS
C - F30602-94-D-0066/0015
PE - N/A
PR - R516
TA - QD
WU - 17

6. AUTHOR(S)
Dr. A.E.R. Woodcock and Dr. Loren Cobb

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Synectics Corporation
10400 Eaton Place
Fairfax VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space Missile Command/XRE
2420 Vela Drive
Los Angeles CA 90245-5500

Rome Laboratory/IRRE
32 Hangar Road
Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-96-242

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Mr. Peter J. Costianes/IRRE/(315) 330-4030

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This final report provides an overview of the functionality of the Space Effectiveness Analysis Simulation (SEAS) System software. It describes a sample use of the SEAS system, the generation of system-generated data, and the use of a statistical software system to analyze the data. Also, presented is a detailed description of TacLan, the tactical language for the SEAS system. More extensive details of the nature and use of the SEAS system are provided in the SEAS User's Manual.

14. SUBJECT TERMS
Space simulation, statistical analysis, effectiveness model

15. NUMBER OF PAGES
72

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

ABSTRACT

This final report provides an overview of the functionality of the Space Effectiveness Analysis Simulation (SEAS) System software. It describes a sample use of the SEAS system, the generation of system-generated data, and the use of a statistical software system to analyze the data. Also presented is a detailed description of TacLan, the tactical language for the SEAS system. More extensive details of the nature and use of the SEAS system are provided in the SEAS User's Manual.

TABLE OF CONTENTS

1.0 THE SPACE EFFECTIVENESS ANALYSIS SIMULATION (SEAS) SYSTEM

1.1 INTRODUCTION

Synectics Corporation is pleased to submit this Final Technical Report for the Space Effectiveness Analysis Simulation (SEAS) system (Contract Number: F30602-94-D-0066/0015) to the United States Air Force, Space Systems Division, Los Angeles Air Force Base. Dr. A.E.R. Woodcock (Synectics Corporation) was SEAS project director. SEAS system software design and implementation were undertaken by Dr. Loren Cobb and Mr. E.H. Pedersen (Corrales Software and Synectics' SEAS consultants).

The Final Technical Report is divided into four major sections:

1. *Section 1.0* provides an overview of the functionality of the SEAS system software.

2. *Section 2.0* describes a sample use of the SEAS system, the generation of system-generated data, and the use of a statistical software system to analyze these data.

3. *Section 3.0* identifies the minimal requirements for the SEAS system and how these requirements have been met by the system that has been delivered to the Air Force.

4. *Section 4.0* provides a detailed description of TacLan, the tactical language for the SEAS system.

More extensive details of the nature and use of the SEAS system are provided in the SEAS User's Manual.

Synectics Corporation would like to acknowledge with gratitude the interest and technical support of Mr. Robert Weber (Aerospace Corporation) in the SEAS project, under what were often difficult circumstances. Synectics Corporation would also like to acknowledge the technical guidance and input provided by the United States and Aerospace Corporation personnel who were involved in the different phases and activities associated with the SEAS project.

1.2 THE SATELLITE EFFECTIVENESS ANALYSIS SIMULATION (SEAS) SYSTEM OVERVIEW

This section of the Final Technical Report provides an introduction to the modeling, analysis, and other methods used in the development and implementation of the Space Effectiveness Analysis (SEAS) system. The SEAS system can be rapidly modified and used to discover relationships between system components, determining the impact of a range of values of military parameters on combat outcome, and identifying emergent system properties. Smaller, flexible models permit the undertaking of a large number of experimental studies while the size, complexity, and inflexibility of scripted models can prevent such studies, and may lead to the production of results that have been pre-determined by the scripting activities.

1.2.1 THE SPACE EFFECTIVENESS ANALYSIS SIMULATION (SEAS) SYSTEM USES MINIMALIST MODELING TECHNIQUES

The Space Effectiveness Analysis Simulation system is a prototype combat modeling and analysis environment. It is intended to be used to assess the impact of the capabilities of modeled satellite and other sensor and communication systems, weapons systems, rules of engagement, and other factors on combat outcomes. The SEAS system was designed to permit its users to define ranges of values such system parameters as satellite orbital characteristics; sensor type, location, range, and detection capabilities; weapon ranges and accuracies; target vulnerabilities; communication system capabilities; combat termination thresholds and rules of engagement and to assess their impact on combat outcomes.

The SEAS system was produced according to *minimalist modeling principles* defined by Dockery and Woodcock (1993). In this way, an attempt was made to construct the simplest possible model of the combat environment involving satellites and other entities so that an extensive series of experimental studies could be carried out to investigate the impact of a range of different factors on combat outcomes. Minimalist modeling activities avoid the construction of detailed scripts where all actions to be carried out are specified in advance. Rather, the overall behavior of such models emerges from the often highly complex and non-linear interactions between the components of the model. In general, the nature of such emergent behaviors cannot be scripted in advance. The characteristics of the individual model components are defined in a relatively simple manner during the model building process.

Other types of combat models do attempt to script many if not all activities in advance and as a result often consist of very large amounts of computer code, are very cumbersome, difficult and time-consuming to modify, and are generally unresponsive to changing user needs. Such models could be characterized as "maximalist" models. Furthermore, these larger models do not usually provide an environment where non-linear, sudden, and unexpected events can occur. While some of these larger models do create variability through the inclusion of random events, in fact, by scripting many, if not all, of the events and activities in advance, such maximalist models have already provided an indication of the outcome even before the model has been run on a computer. In the case of the SEAS system, combat-related behavior is generated as the result of the interaction of entities who are responding to a limited set of instructions, not an extensive script.

The heart of the SEAS system is the simulation engine in which many of the sensor- and weapons-related components have been implemented using a range of mathematical modeling techniques including a method which uses intelligent automata (Figure 1-1). Intelligent automata-based modeling has been developed by Cobb and Woodcock in order to permit an analysis of the time- and space-dependent behavior of the combat environment (see: Woodcock, Cobb, and Dockery, 1989, and Dockery and Woodcock, 1993, for example). In addition, other mathematical and computer modeling techniques have been used to create the SEAS environment within which assessment of the impact of satellite characteristics and other factors on combat outcomes can be assessed.

Intelligent automata entities are provided with a series of properties and rules which permit them to undertake an analysis of those entities that are in their immediate neighborhood, or that are otherwise able to influence their behavior, assess the overall situation, and the react according to a set of rules of behavior or engagement. Activities occur in different locations and can be the result of time-base- or event-driven influences. The behavior of an ensemble of automata entities is based on the interaction of individual entities and the overall behavior is characterized as an emergent property of the system of entities. The overall behavior is seldom, if ever, predictable from the nature of the rules and other properties assigned to individual automata entities.

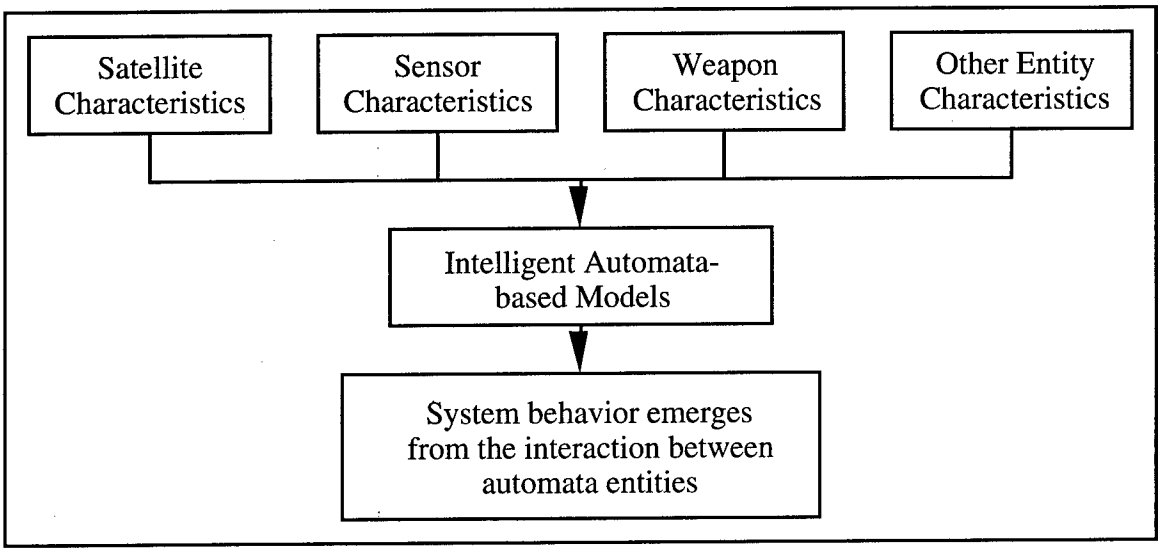


Figure 1-1: Overall SEAS system behavior is generated from the interaction of many simple system components.

1.2.2 FUNCTIONAL RELATIONSHIPS ASSOCIATED WITH THE SEAS SYSTEM

The SEAS system provides facilities for assessing the impact of a range of user-selected factors on combat outcomes. Behavior of the modeled combat entities is calculated on the basis of input data with the aid of the models contained within the SEAS simulation engine. Data input is coordinated within the SEAS Deployment Editor facility and made available for use by the SEAS simulation engine. An overview of the functional relationships within the SEAS system and the nature of the system inputs and outputs is presented in Figure 1-2. Details of the operation of these facilities during use of the SEAS system is described in subsequent parts of this manual.

1. DATA INPUT to the SEAS system is provided directly by the user and from other satellite model-based and Defense Mapping Agency product sources.

a. *User-selected data* can be entered with the aid of menus and windows provided within the overall deployment editor facility.

b. *Satellite-related data* is provided by the Satellite Orbital Analysis System (PC-SOAP) developed by Aerospace Corporation and used in conjunction with the SEAS system. The PC-SOAP system calculates satellite orbital trajectories and provides the SEAS system with information on the timing, pattern, and nature of satellite coverage and communications capabilities for selected areas of interest to the SEAS user.

c. *Map-related data* for the SEAS system is provided electronically from selected Defence Mapping Agency map products, as described in more detail below.

2. DATA ENTRY, ASSEMBLY, AND TRANSFER to the SEAS simulation engine is accomplished with the aid of the SEAS Deployment Editor and is made possible through the use of the SEAS Deployment File. These data are translated into a deployment file that is implemented in TacLan, a high-level computer language

developed to support the creation of deployment data files for use in conjunction with models based on intelligent automata. This facility and other components of the SEAS system utilize an extensive graphics interface capability.

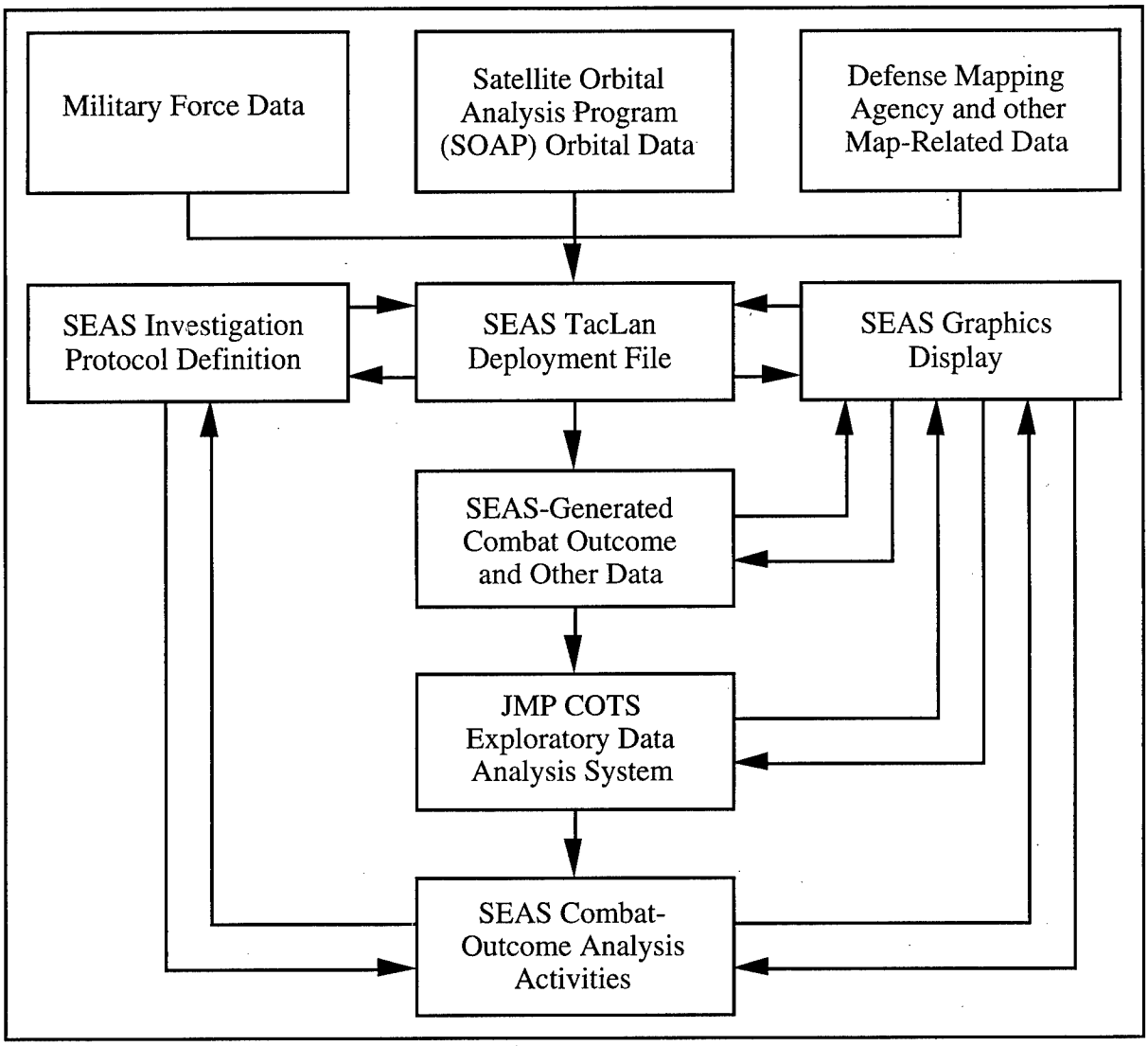


Figure 1-2: Overall functional relationships within the SEAS system.

3. SEAS INVESTIGATION PROTOCOL DEFINITION takes place within the Deployment Editor facility and permits the user to define the nature of the investigations that are to be carried out within the SEAS system.

4. COMBAT OUTCOME MODELING is performed by the SEAS Simulation Engine and the results of these activities depends on the nature of the input data provided through the Deployment Editor.

5. DATA COLLECTION AND STATISTICAL ANALYSIS is performed on data generated with by the Simulation Engine and stored within the SEAS system. Statistical analysis of these data are carried out with the aid of the commercially available

JMP statistical software system produced by the S.A.S. Institute. Results of these analytic activities can be used to define additional studies.

1.2.3 COMBAT MODELING ACTIVITIES WITHIN THE SEAS SYSTEM

The overall nature of the combat modeling activities undertaken within the SEAS system is shown in Figure 1-3 involves the transformation of the user-defined combat environment into models within the simulation engine and the subsequent use of these models to compute combat outcomes.

1. THE DEPLOYMENT EDITOR GENERATES COMBAT-TRUTH DATA that are translated into the TacLan language and made available to the Simulation Engine in the form of the SEAS Deployment File.

2. THE SIMULATION ENGINE CREATES AND RUNS A COMBAT MODEL based on the input from the Deployment Editor. The combat modeling activities supported by the Simulation Engine are outlined below for a fictitious Blue Force modeled within the SEAS system, but apply equally well to the Red Force.

a. *Sensor Detection* activities involve the use of the modeled systems to detect other modeled combat entities. Entities are initially located with the aid of the Deployment Editor and their subsequent movement and activities take place according to their selected rules of behavior. Sensor detection of entities within the modeled SEAS combat environment will occur with a probability based on the user-defined characteristics of the selected sensor systems.

b. *Situation Assessment* activities involve the use of Intelligent Automata-based entities known as Situation Appraisal Agents (SAAs). These Agents utilize sensor-derived and other data to assess the level of threat posed to the modeled friendly force entities by adversarial entities.

   i. Entity Location depends upon the nature of the sensors involved and the timeliness of the reporting of sensor-derived information to the Appraisal Agents. Thus, while a sensor might provide an initially relatively precise location based on the inherent level of accuracy of the sensor device for an entity (at time T_1, Figure 1-4, for example), the possible area within which the entity could be located increases in size as time progresses and the transmission of data to the Appraisal Agent is delayed (at time T_1 and T_1, for example). The size of the area of uncertainty is obviously a function of the maximum speed at which a detected entity could move, Thus the area of uncertainty would be relatively small for a tank (Figure 1-5a), and increasingly larger fro a helicopter and an airplane (Figures 1-5b and 1-5c) for the same elapsed time interval. The Situation Appraisal Agents are also involved in threat map calculation, as described below

   ii. Threat Map Production activities are performed by the Situation Appraisal Agent entities and these maps are used to provide the command entity (referred to as a Command Agent, below) with information that can be used for the command and control of modeled subordinate forces. The process for threat map production used within the SEAS system, illustrated in Figure 1-6, is described below. An approach (technically known as cluster analysis) is used to calculate the size of an elliptical area

that is occupied by at least 80 per cent of the ground force components of an adversarial force. The process then calculates the shape of an entity, called the *Ground Force Threat Map*, that is in the shape of a rectangle with edges that are equal to the length and width of this ellipse. The shape and position of the Threat Map rectangle are calculated and updated at each time-step of the simulation engine. The location of the Threat Map is used to guide the decision-making activities of the command entity as the modeled combat engagement proceeds

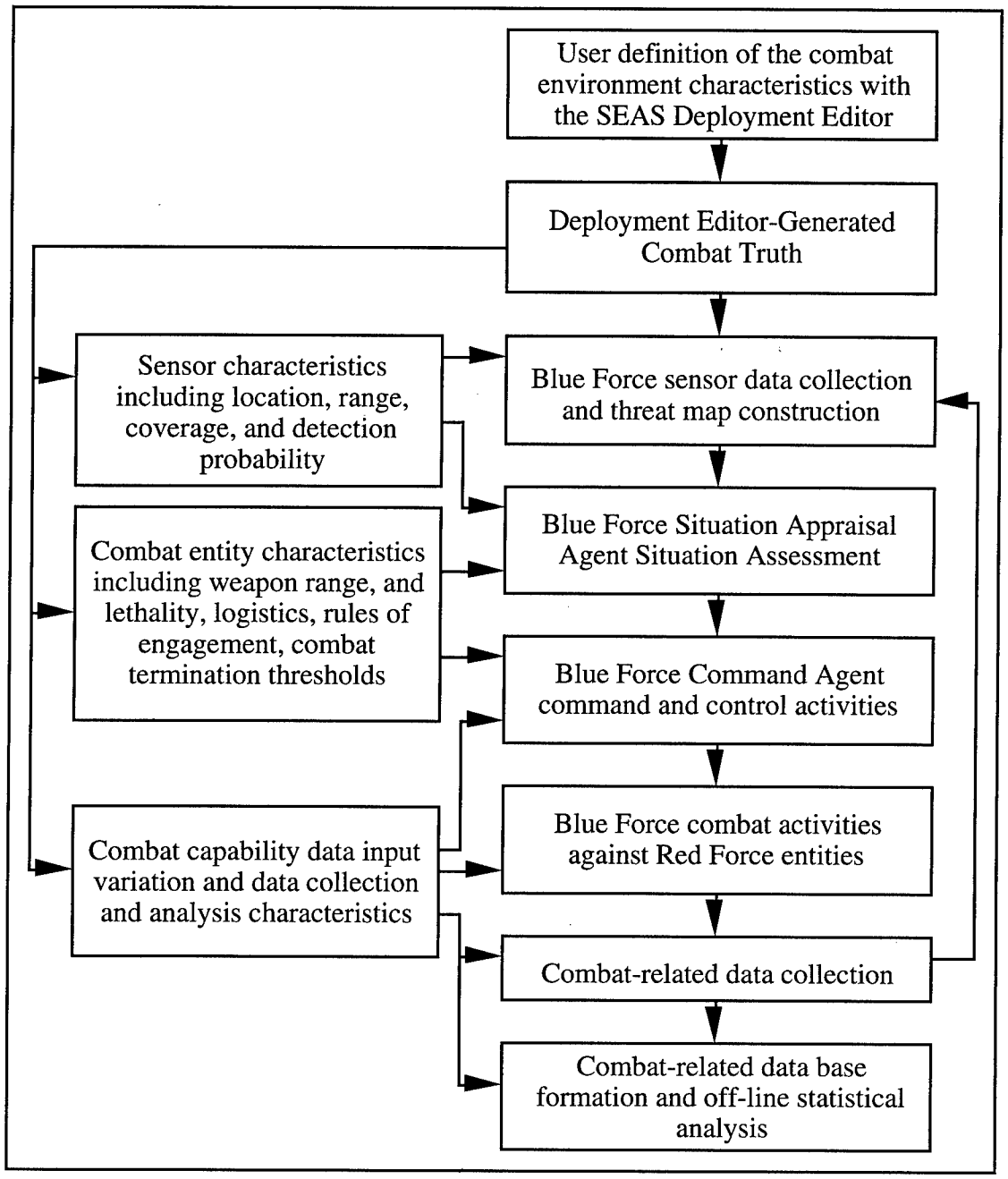


Figure 1-3: Selected combat modeling activities supported by the SEAS system.

c. *Command Agent* activities involve the use of Intelligent Automata-based entities known as Command Agents (CAs) that undertake decision-making activities involved in the command and control of subordinate forces. Orders issued by the Command Agents generally lead to the deployment of forces and their engagement with hostile entities in some form of combat activity. Combat activities generally lead to notional casualties and the levels of these casualties are reported to the Command Agents (Figure 1-7). The Command Agent could order a force under its command to cease attacking and retreat when the level of casualties exceeded a user-defined combat termination threshold. Thus, while the Blue force Command Agent in Figure 1-7a, would have permitted combat to continue, the Red Force Agent could have issued an order to cease fighting at time T_1 (as shown in Figure 1-7b, for example).

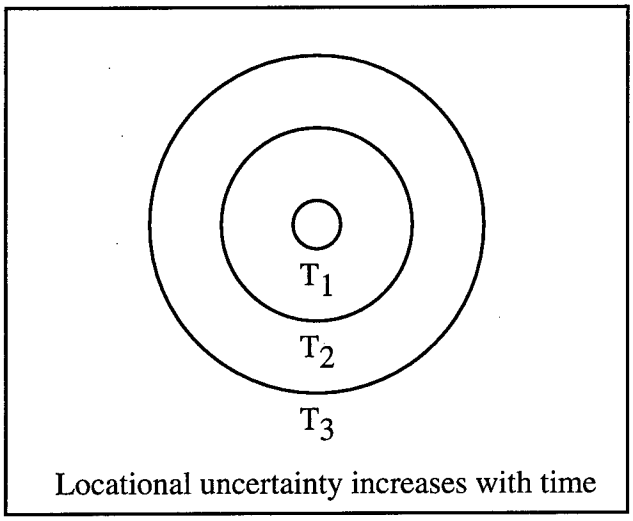


Figure 1-4: Delays between the initial detection of an entity by a modeled sensor system and the reporting of this information to a Situation Appraisal Agent can increase the uncertainty of the location of the detected entity.

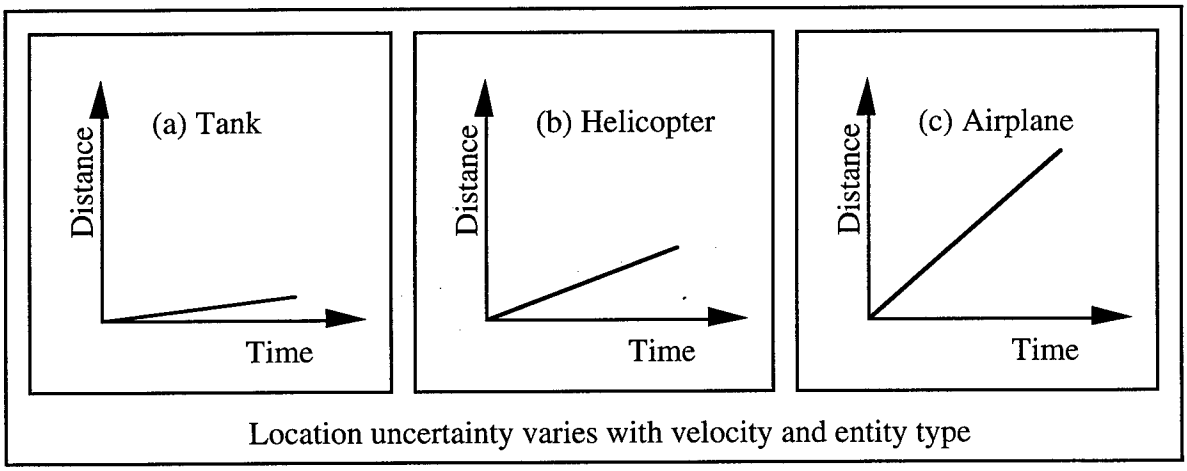


Figure 1-5: The location uncertainty for an entity depends upon the maximum velocity with which the entity can move. Thus, for a given elapsed time interval, the location uncertainty would be increasingly greater for (a) a tank, (b) a helicopter, and (c) an airplane.

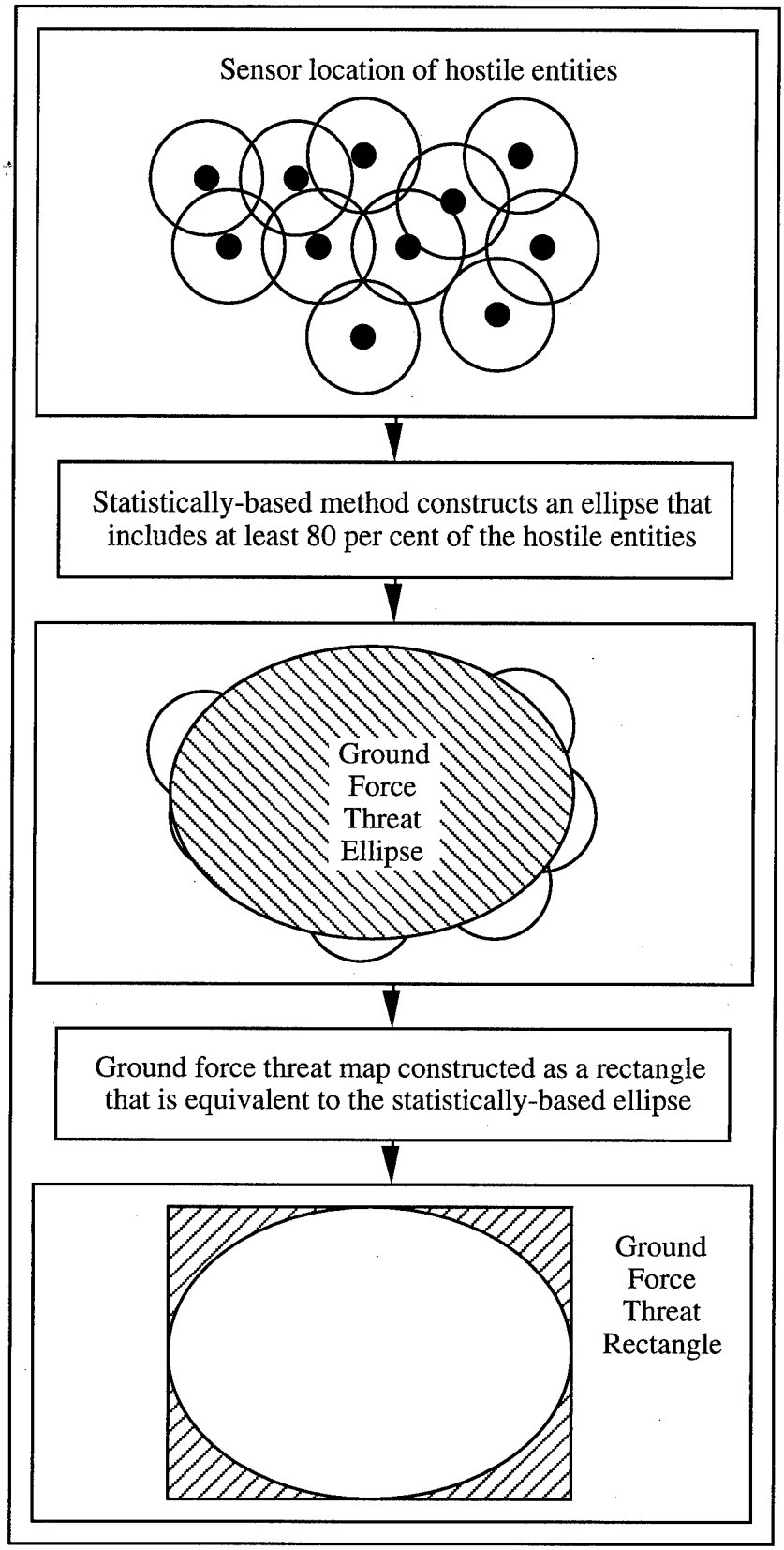


Figure 1-6: Situation Appraisal Agents calculate the location of ground threat maps.

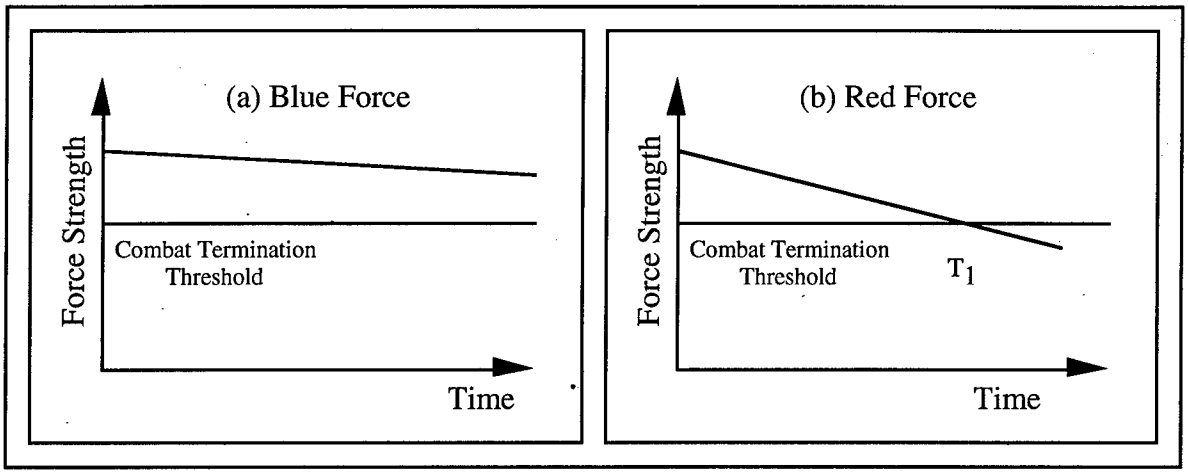


Figure 1-7: Combat casualty data and other information is used by the Command Agent entities to support the command and control of subservient forces.

d. *Data Collection and Off-line Statistical Analysis* facilities are provided within the SEAS environment and permit users of the system to undertake a wide range of statistical investigations that can provide new types of insight into the nature of the modeled combat environment. Such studies could be used to examine the effectiveness of different types of satellite-based systems, air and ground-based sensors, weapon systems, and strategies, tactics, and rules of engagement.

1.2.4 SEAS SOFTWARE COMPONENTS

The relationship between the major software components of the SEAS system are illustrated in Figure 1-8. As mentioned above, the SEAS Deployment File is the data input source for the SEAS system. User-generated input is accomplished with the aid of the Access Database and SEAS Deployment Editor facility. Data from the Satellite Orbital Analysis Program (PC-SOAP) is made available to the SEAS system after its conversion with the READSOAP.EXE conversion tool facility. Map-based data from Defense Mapping Agency sources is converted with the VPF.EXE raster conversion tool facility.

Information from the SEAS Deployment File forms basis of the WAR.EXE file and is used to drive the SEAS Simulation Engine. This file provides a representation of the nature and capabilities of both allied (friendly) and enemy forces in terms of program objects that are created for manipulation by the simulation engine. Entity relationships are illustrated in Figure 1-8, and include the following: both allied and enemy forces can possess units and satellites; units can own airplanes, other vehicles, weapons systems, communication channels and sensors; satellites can own sensors, communication channels, and weapon systems; and airplanes can own sensors, communication channels, and weapon systems.

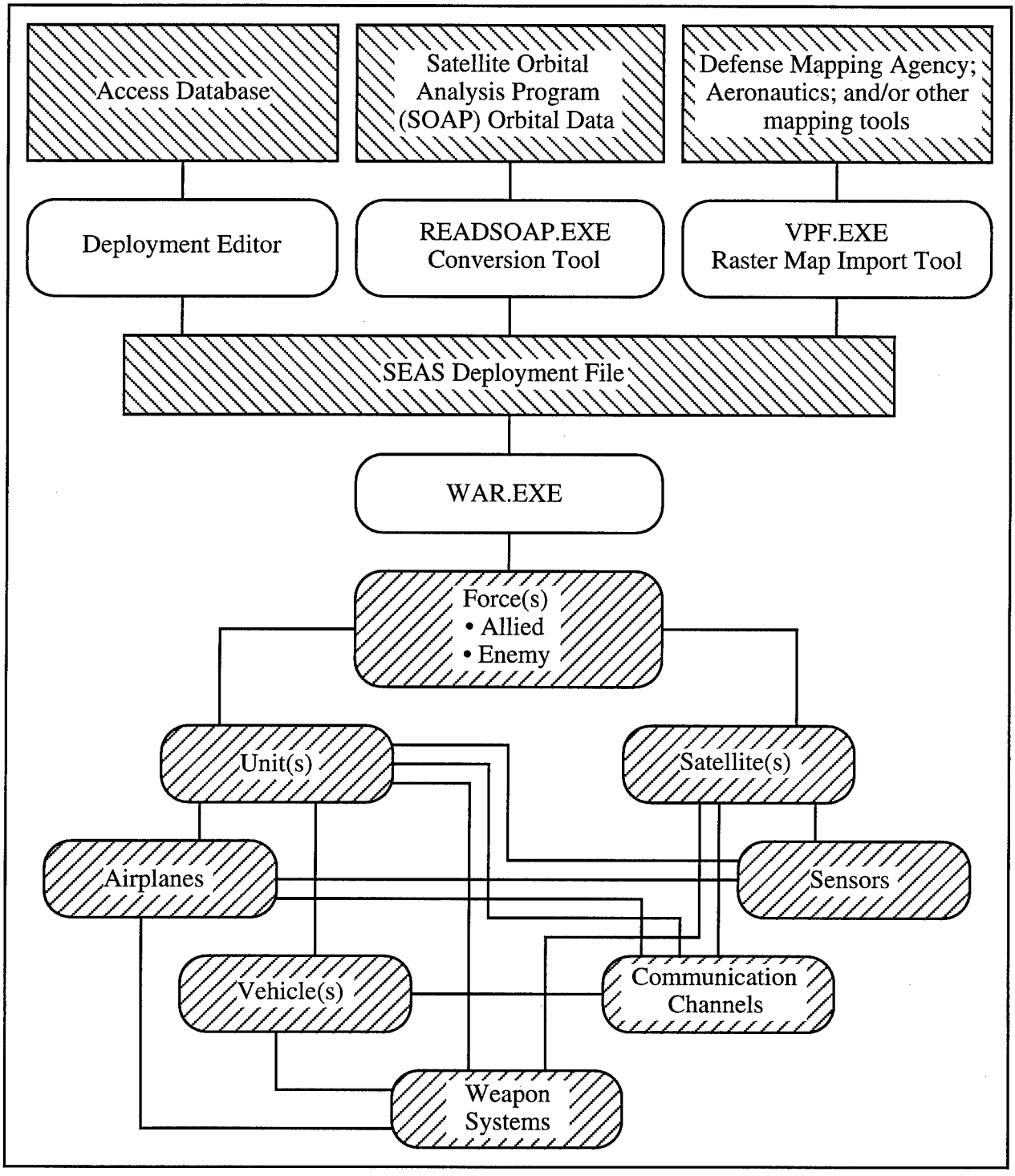


Figure 1-8: Relationships with the overall SEAS software environment.

2.0 USING THE SEAS SYSTEM

This section of the Final Technical Report describes activities involved in running the SEAS system and provides a report of the use of the SEAS system to model a simple user-defined combat scenario.

2.1 SEAS DEPLOYMENT EDITOR CAPABILITIES

The SEAS Deployment Editor permits the user to define parameters of the different components of the modeled combat environment with the aid of a series of graphic interface facilities. Output from the Deployment Editor is used to generate a TacLan Deployment File for the SEAS system (Figure 2-1). The Deployment Editor also provides a facility for incorporating map and satellite orbital analysis data into the overall SEAS environment. Map-based data are generated as an output from the VPF map conversion tool and the orbital data is derived from the Satellite Orbital Analysis Program (PC-SOAP) log conversion tool as outlined in the SEAS User's Manual. The TacLan-based output from the deployment editor is used to define the nature of the military environment under investigation by the user of the SEAS system.

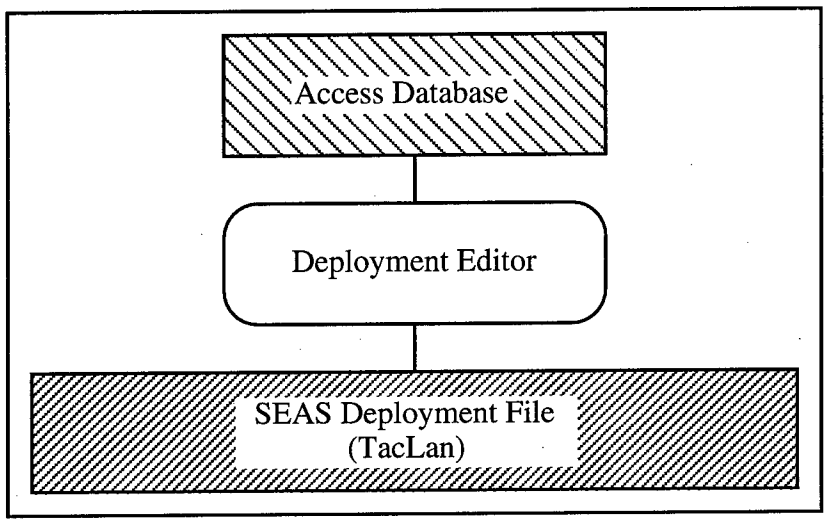


Figure 2-1: The SEAS Deployment Editor permits the entry of user-selected data and provides output which is used to create the TacLan-based SEAS Deployment File.

Specifically, the Deployment Editor provides an ability to define and modify the characteristics of the following model components: Tactical Area of Operations (TAOs), Platforms, Time Line, Locations, Forces, Plans, Airplanes, Units, Communications, Weapons, Sensors, Satellites, Symbols, and Targets by means of software buttons (Figure 2-2). Details of the types of user-entered data that could be made available to the SEAS environment for modeling and analysis purposes are described in more detail below.

- *Tactical Area of Operations (TAOs)* data defines the map-based area within which a force or unit is assigned operational responsibility.
- *Platforms* data defines the nature of the weapons, sensor, and other types of platforms to be deployed within the SEAS environment.

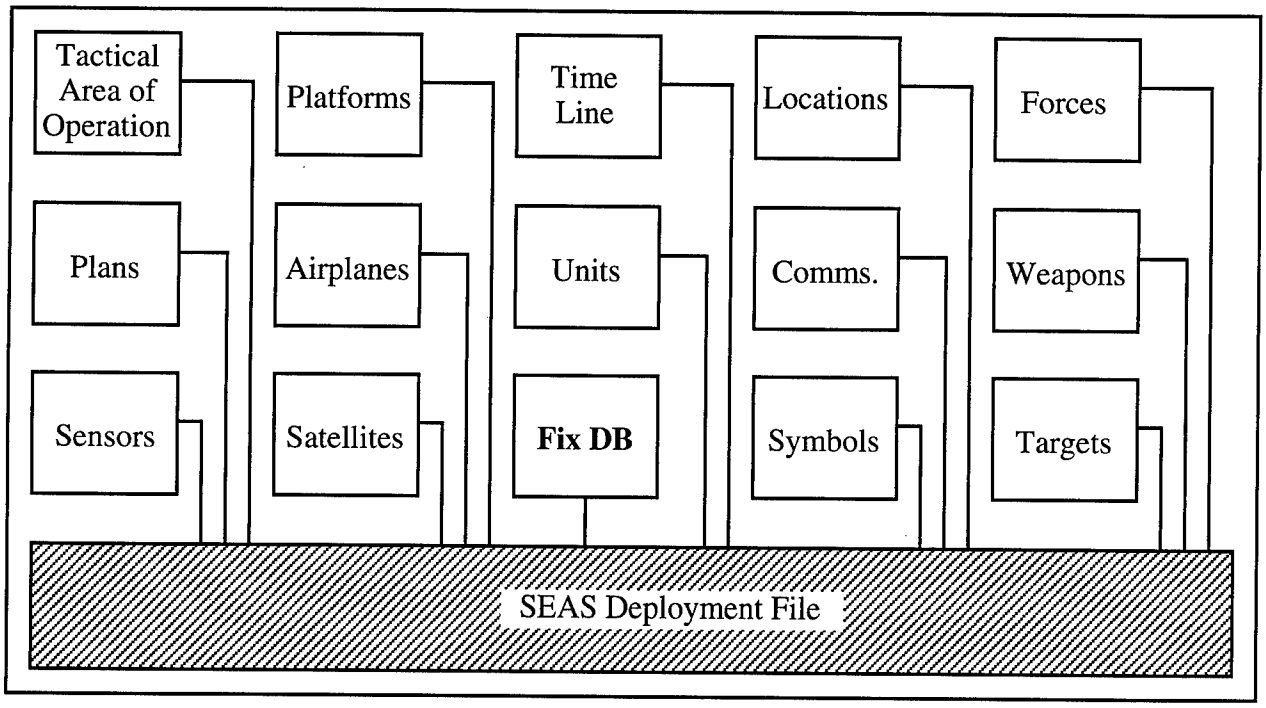


Figure 2-2: The tab structures in the graphics interface permits the user of the SEAS system to define operational and equipment aspects of the modeled combat environment.

- *Time Line* data provides information on the duration of the overall modeling activity and the times at which events can occur within this time.
- *Locations* data provided information on the deployment and movement position of the entities modeled within the SEAS environment.
- *Forces* data describes the nature of the friendly and enemy forces deployed within the SEAS environment as well as the major sub-components of these forces.
- *Plans* data provides information on the user-defined activities to be undertaken be friendly and enemy forces.
- *Airplanes* data provides information on the number, type, weapons, sensors, and operational characteristics of aircraft deployed within the SEAS environment.
- *Units* data provides information on the composition and nature of the different units deployed within the SEAS environment.
- *Communications* data provides information on the operational characteristics of the SEAS communication systems.
- *Weapons* data provides information on the range, accuracy, and lethality of the SEAS weapons entities.
- *Sensors* data provides information on the range, accuracy, and other operational characteristics of the SEAS sensor systems.
- *Satellites* data provides information on the operational characteristics of the SEAS sensors and the types of facilities that the satellites can provide.

- *Symbols* data provides access to the icons and symbols that can be used to represent different entities deployed within the SEAS environment.
- *Targets* data provides information on the location, nature, vulnerability, and other characteristics that serve as targets within the SEAS system.
- The "Fix DB" button is used only during development to insert new fields into an old deployment so that the old deployment does not have to be recreated from scratch.

2.2 CREATING A SEAS TACLAN FILE

The model component data elements described above can be defined in detail by the SEAS user. These data elements can then be translated by the SEAS system to create a TacLan-based SEAS Deployment File which serves as the input to the Simulation Engine facility. The Simulation Engine is located in the SEAS\BIN32 directory with the filename WAR.EXE and is used to process the data contained in the SEAS Deployment File (Figure 2-3).

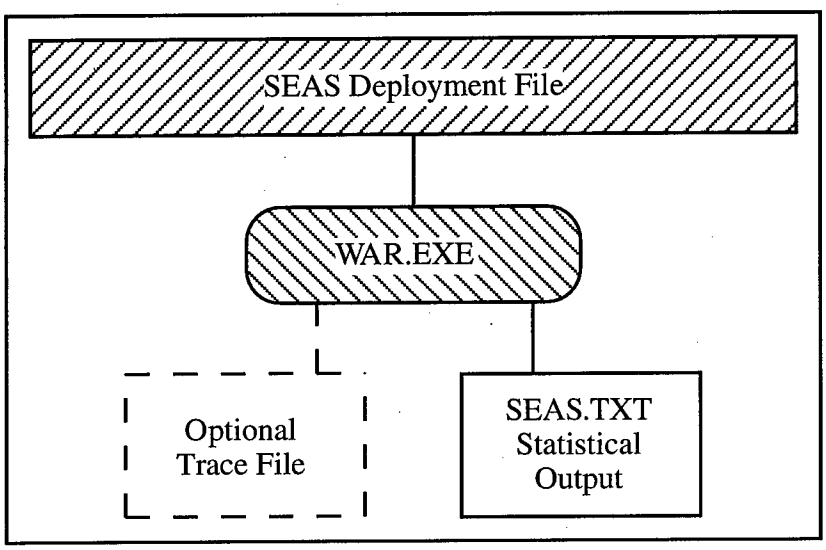


Figure 2-3: The SEAS Deployment File provides input data that can be used to drive the SEAS Simulation Engine facility.

2.3 SEAS MAP AND FORCE STATUS DISPLAYS

The maps selected with the aid of the SEAS Deployment Editor facility can serve as a background upon which the user-defined combat simulation can take place. The user defined input properties of the simulation are used by the SEAS simulation engine to generate dynamical combat behavior. The user is provided with a number of display options for maps and for the modeled forces engaged in combat. It is possible for the user to display an overall world map, a more localized regional map, and/or a local map.

- *The world map display* permits the user to observe the movement of satellites, the pattern of day/night transitions, and the location of the defined battlefield.

- *The regional map display* permits the user to observe the movement and interaction of forces within a specified operational-like environment.

- *The local map display* permits the user to view a close-up version of the battlefield. Deployment designers may find it useful to substitute a perfectly blank map for the local map, so that units can be viewed without distractions from the background. Such a blank display has been used below.

The map displays can show the position of the day/night terminator and projections onto the map of the locations of the orbits of two user-defined satellite entities. The display also provided the user with access to data concerning the status of (a) All (Blue and Red) Forces, (b) Blue Forces, or (c) Red Forces. User selection of one of these three options provides access to the relevant data and to an appropriate map display.

Selection of an All Force option by using the "Perspective" window will cause the system to display the composite data for both Blue and Red Forces as well as a map that shows the location of both forces and the assets under their control. Selection of the Red Force option, by contrast will permit the display of Red Force data and a map which shows the location of assets that are under control of the Red Force only.

The map display facility provides the following information elements that reflect the status of the adversarial forces.

- Date and time.
- Elapsed time in the simulation run.
- The number of the specific run in the user-defined sequence of runs.
- The number of troops involved.
- The number of casualties.
- The number of units involved in the combat activity.
- The number of vehicles involved in the combat activity.
- The number of weapons involved in the combat activity.
- The number of planes involved in the combat activity.
- The number of messages that are waiting in the queue for the relevant communications channel.

Data generated during each simulation run are stored in the system and can be used for statistical analysis after the modeled activities have ceased, as outlined below.

2.4 SAMPLE USE OF THE SEAS SYSTEM

The following is a brief description of the sue of the SEAS system to examine the impact of several combat variables on combat outcome (Figure 2-4). In this case a user has used the SEAS Deployment Editor to define a simple engagement between relatively small Blue and Red forces,

After the production of a TacLan file, the system has been used to undertake several runs of the system and data has been collected and analyzed statistically with the aid of the JMP Exploratory Data Analysis system.

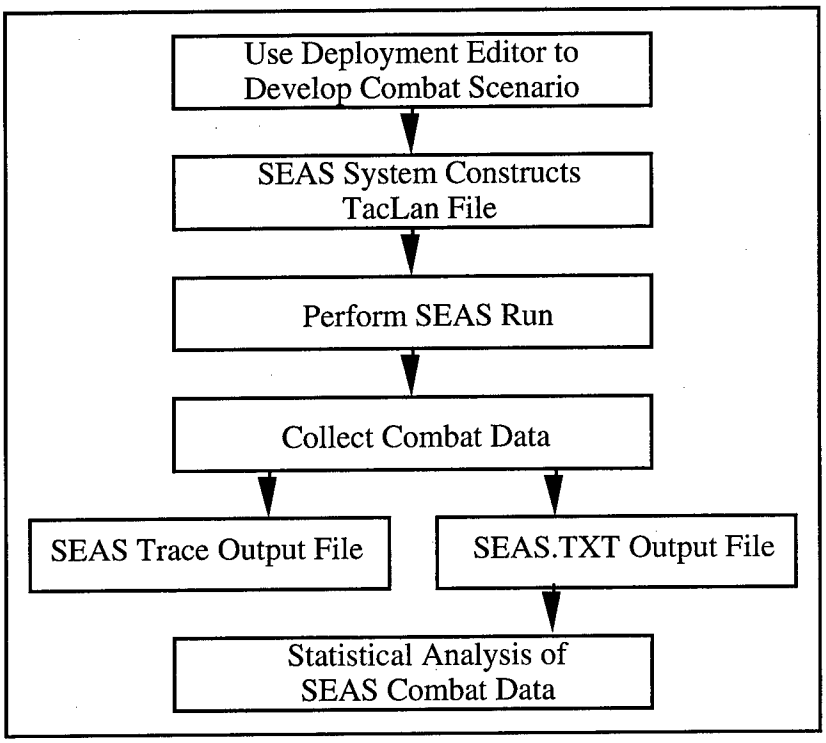


Figure 2-4: Activities involved in using the SEAS system to examine the impact of user-selected parameter values on combat outcomes.

Screen-dumps involving blank map displays that collected during a sample use of the SEAS system are shown in Figures 2-5 to 2-7. Figure 2-5 shows the combat conditions at 08.46 hours after combat began at 06.00 hours. The black bar in the upper left-hand side of the figure is a pair of cruise missiles moving from the left. The hatched area in the upper right-hand side of the display is the last Red Force position known to the Blue Force. Red is moving from the Red Base to the Desert Oasis. Blue is trying to make contact with the Red Force.

Figure 2-6 shows modeled combat activities at 09.12 hours after combat initiation at 06.00 hours. The threat rectangles drawn in this figure indicate that both Red and Blue Forces have detected each other. Sensor-derived information provided to Blue has led to the assumption that some Red Force elements are still located close the Red base, but other elements are assumed to be located close to, or at, the Desert Oasis. Simulated shots have been fired, and the Red Force has taken some casualties. As the modeled combat engagement has proceeded to 11.27 hours, the Red force has suffered additional casualties (Figure 2-7). The Blue Force is moving in for the final attack and the weakened Red Force is Retreating to the west.

After the user-defined set of modeled combat engagements have been completed, the data is stored for future analysis, the SAS-JMP exploratory data analysis system is used to perform a statistical analysis of these data. The user can use the "Import" menu item (not the "Open" item) under the "File" menu. the import dialog box is shown in Figure 2-8. The user should make sure that the "Labels" and "Commas" checkboxes are checked and the "Tab" and "Spaces" checkboxes are not checked. A typical SAS-JMP data window is shown in Figure 2-9. The user

is encouraged to edit the column labels since the labels supplied by the SEAS system are generally much too long to fit.

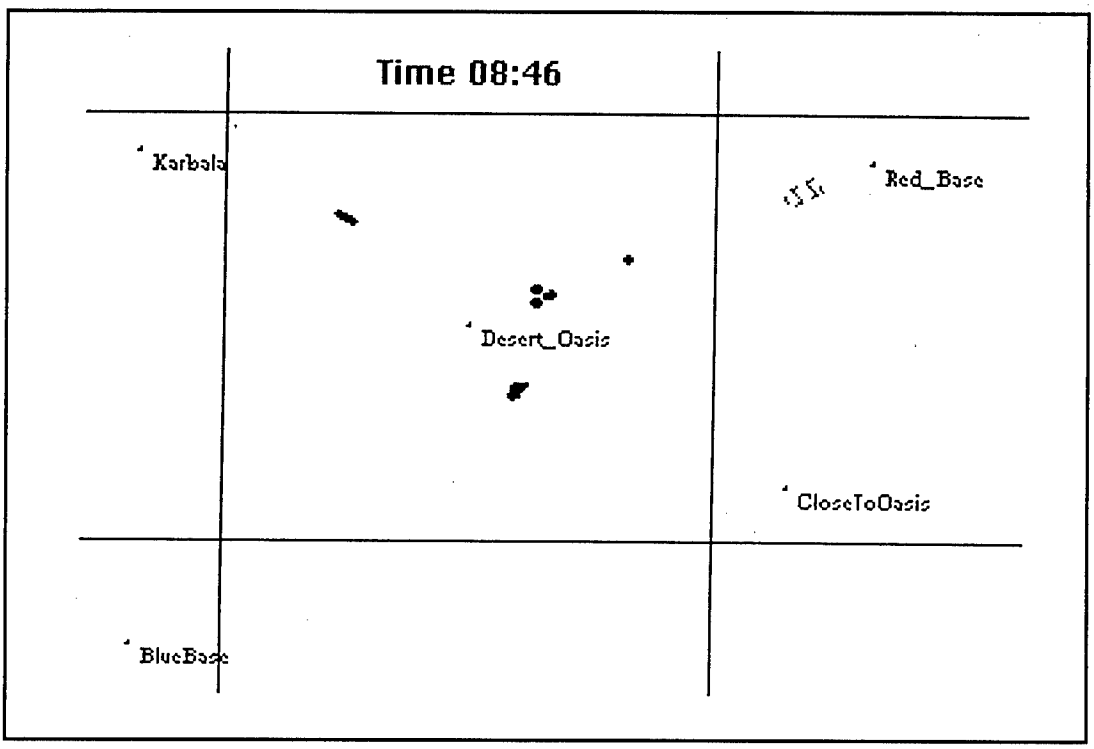


Figure 2-5: Combat movement at 08.46 hours after combat initiation at 06.00 hours.

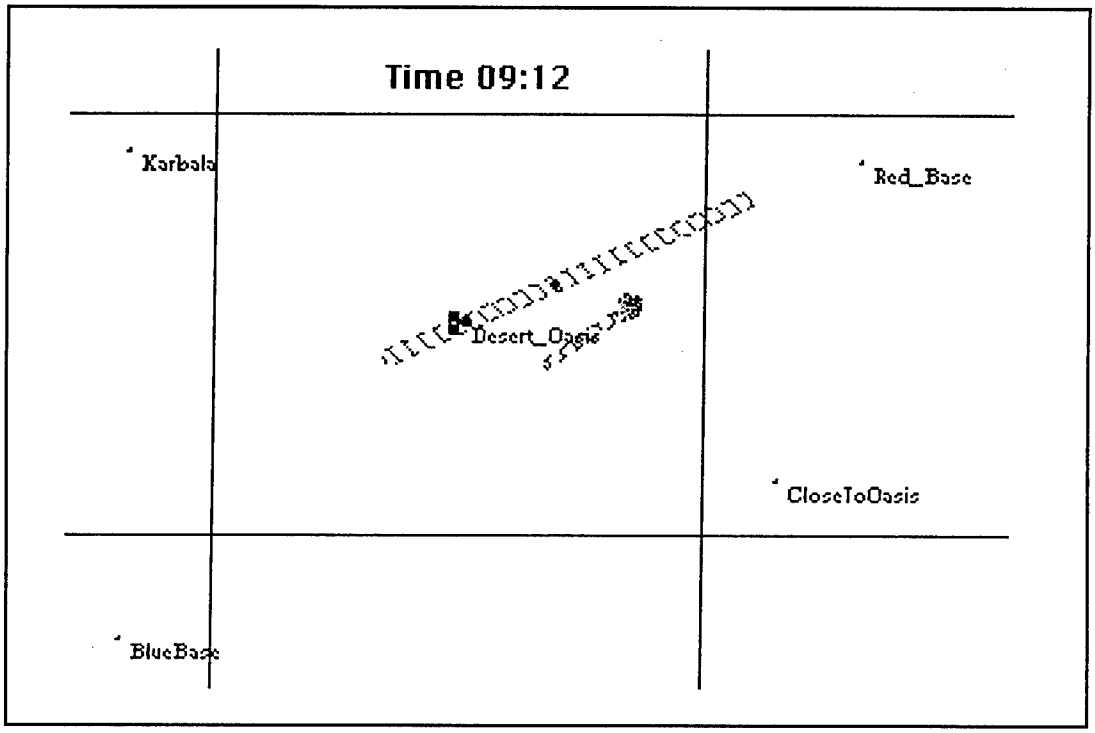


Figure 2-6: Combat activities at 09.12 hours after combat initiation at 06.00 hours.

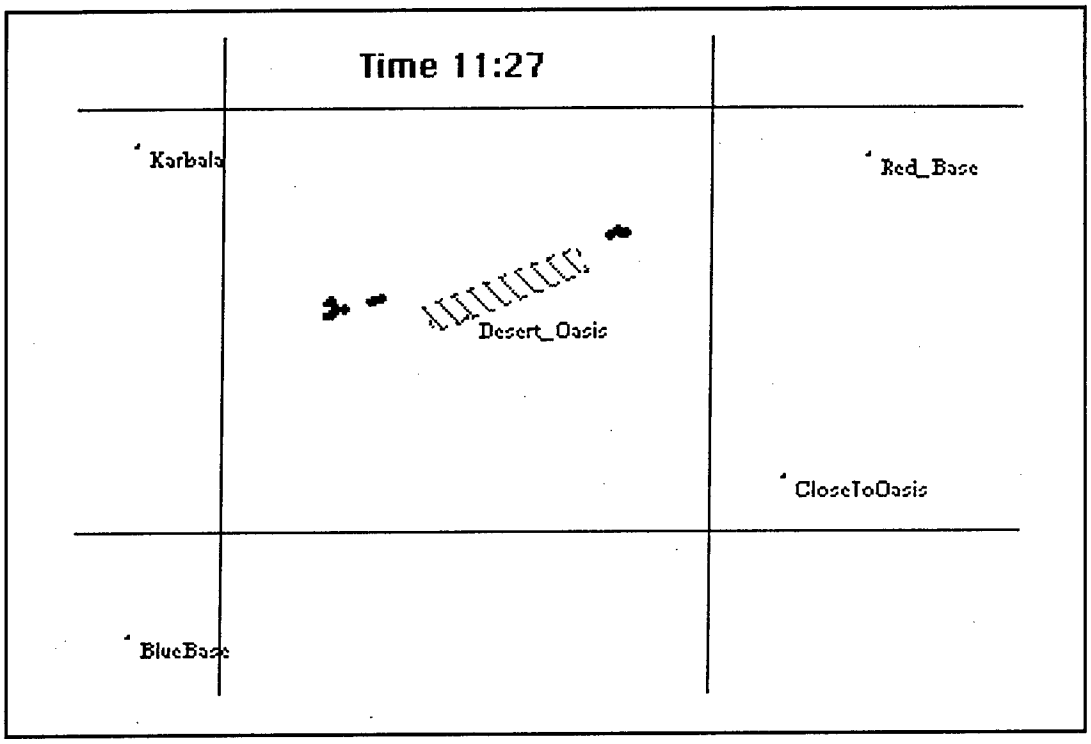


Figure 2-7: Combat activities at 11.27 hours after combat initiation at 06.00 hours.

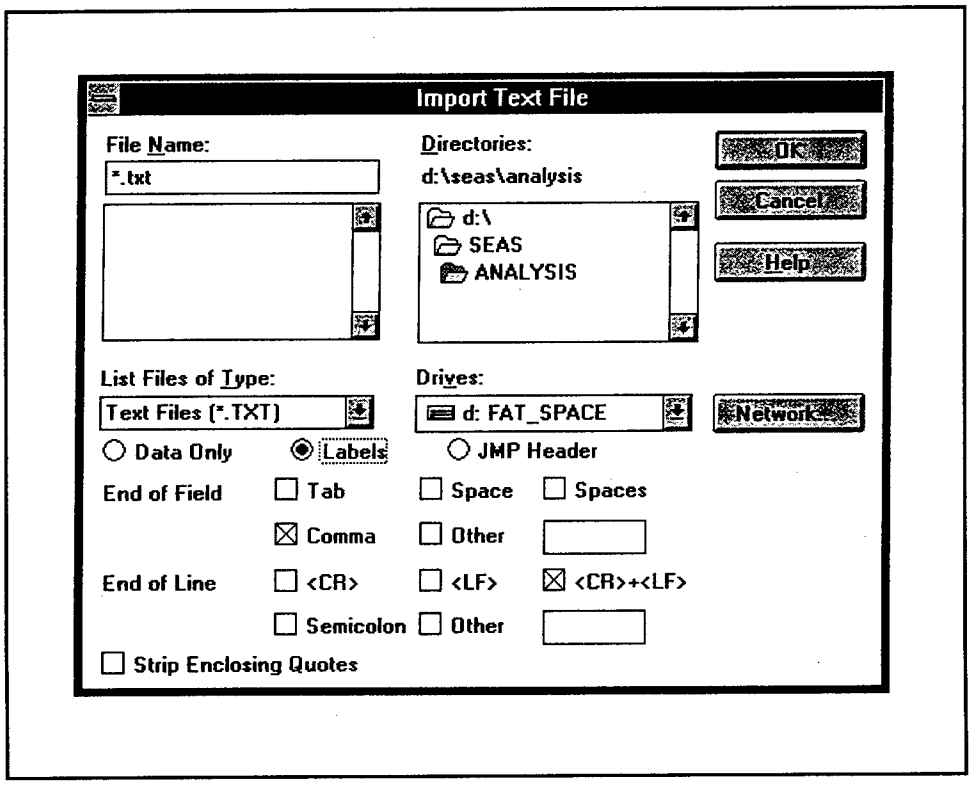
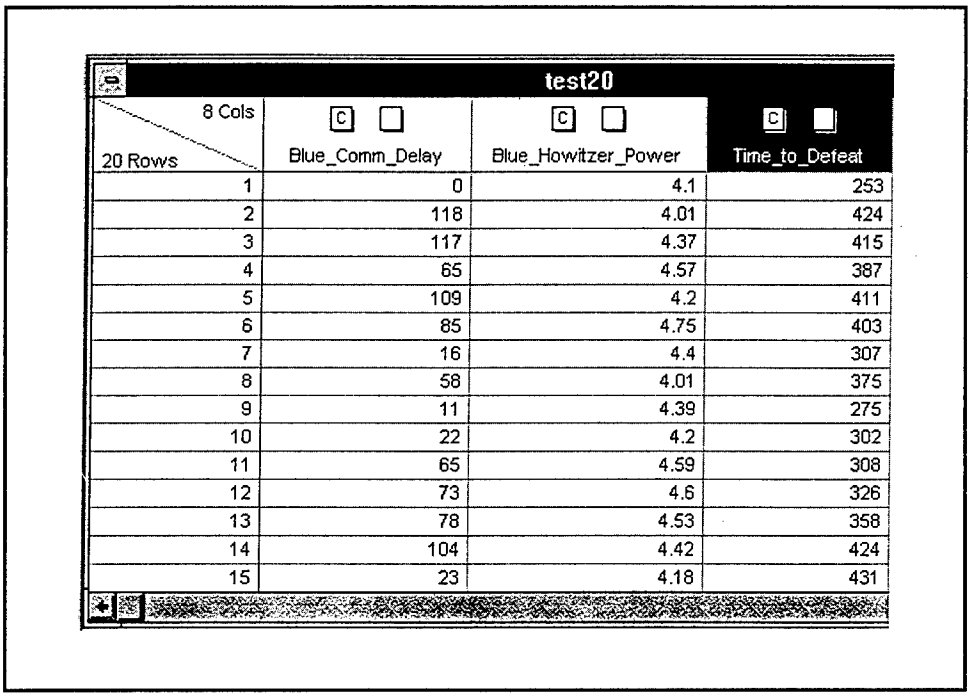


Figure 2-8: The "Import Text File" dialog box permits the transfer of data to the JMP statistical analysis system.



20 Rows	Blue_Comm_Delay	Blue_Howitzer_Power	Time_to_Defeat
1	0	4.1	253
2	118	4.01	424
3	117	4.37	415
4	65	4.57	387
5	109	4.2	411
6	85	4.75	403
7	16	4.4	307
8	58	4.01	375
9	11	4.39	275
10	22	4.2	302
11	65	4.59	308
12	73	4.6	326
13	78	4.53	358
14	104	4.42	424
15	23	4.18	431

Figure 2-9: A typical SAS-JMP data window.

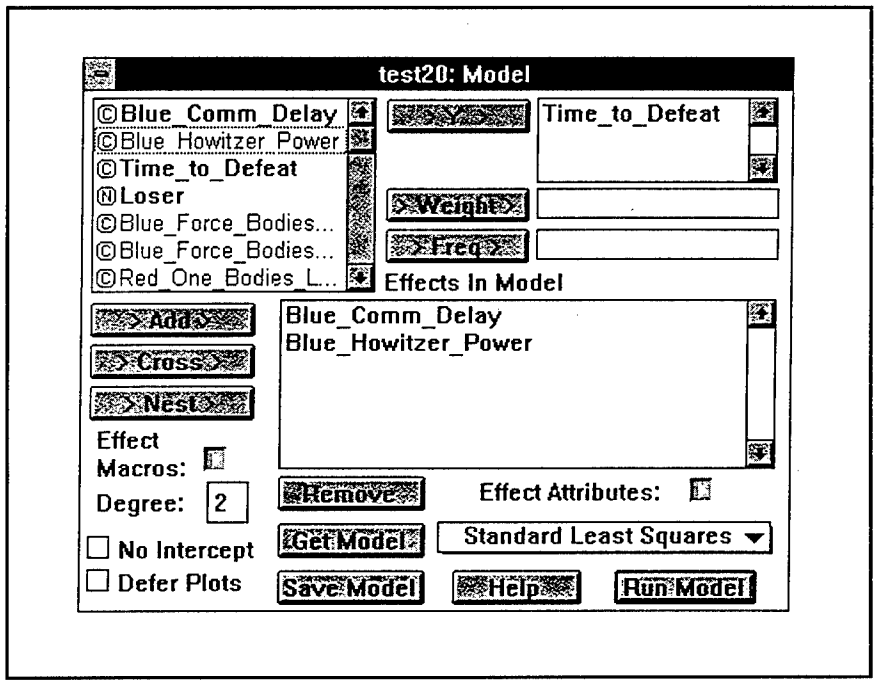


Figure 2-10: Selecting variables for analysis with the SAS-JMP system.

When there is more than one experimentally manipulated variable, then the user should use the "Fit Model" item under the "Analysis" menu. A typical model dialog box is shown in Figure 2-10. The user should make sure that every manipulated variable is included in the box labeled

"Effects in Model," and the dependent variable is located in the upper right-hand corner of the dialog box. The model fitting routine produces graphs similar to that shown in Figure 2-11.

The diagonal line is the statistically-estimated line; the diagonal curved lines provide an envelope of 95 per cent confidence around the estimated line. The "Prob>F" statistic gives the statistical significance of the graphed effect. In this case, the significance value of 0.00005 indicates that the Blue Force satellite communication delay (which was varied between 30 and 120 minutes in the investigation) is shown to be extremely significant.

By contrast, the same analysis shows that the effect of the Blue Force howitzer lethality has no perceptible effect on the time that it takes the Blue Force to Defeat the Red Force (Figure 2-12. This result has been obtained with a modeled combat engagement involving a relatively small number of military entities and 20 replications.

More elaborate simulations with larger numbers and different types of entity and more replications could generate a richer data set that could be subjected to statistical data analyses with the SAS-JMP system.

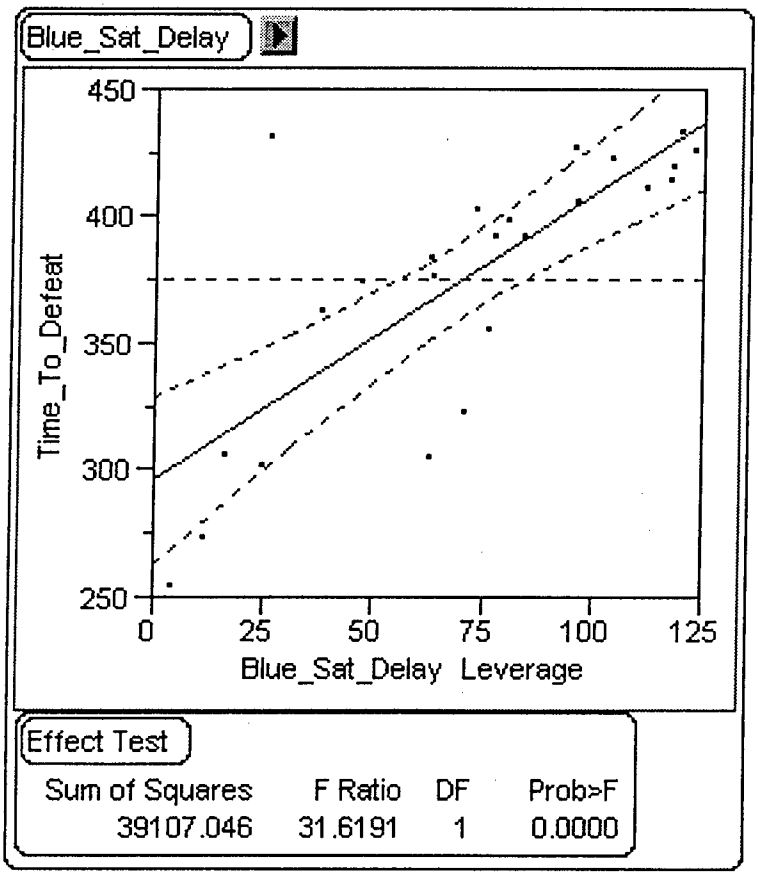


Figure 3-24: SAS-JMP system output showing a statistically significant relationship.

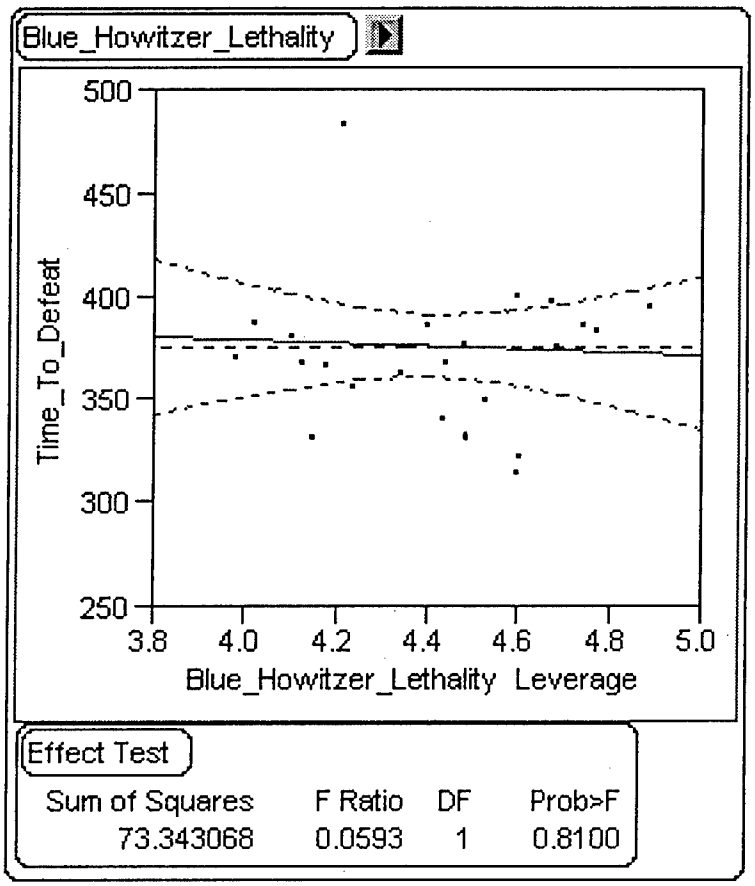


Figure 3-25: SAS-JMP system output showing a statistically insignificant relationship.

2.5 SEAS.TXT OUTPUT FILE

A sample SEAS.TXT output file generated by the system during a particular run is shown below. The general format of the file is one row of column descriptions in text followed by nnn lines of data, one line per iteration run. The contents of the column description will vary in a manner determined by the user. In the example the column descriptions are quite lengthy. They have been "wrapped" into the first five lines of text. The format is acceptable to Lotus 123, the SAS JMP system and other analytical tools. It is, however, beyond the scope of this manual to discuss statistical analysis or the meaning of the SEAS.TXT output values.

Satellite_Leo_Delay,Satellite_Leo_Probability,BlueForce_Casualties, BlueForce_Collat_Comm,
BlueForce_Collat_Weapon, BlueForce_Collat_Vehicles, BlueForce_Collat_Bodies,
RedForce_Casualties, RedForce_Collat_Comm, RedForce_Collat_Weapon,
RedForce_Collat_Vehicles, RedForce_Collat_Bodies, BlueRockets_Casualties,
BlueRockets_Collat_Comm, BlueRockets_Collat_Weapon, BlueRockets_Collat_Vehicles,
BlueRockets_Collat_Bodies, Time_To_Withdrawl_Or_Defeat
397, 6.13, 12.87, 17.25, 20.00, 134, 10.05, 5.59, 0.00, 2.00, 0, 0.00, 0.00, 0.00, 0.00, 278
397, 0.60, 9.25, 43.42, 39.00, 199, 5.43, 0.00, 2.40, 43.00, 0, 0.00, 0.00, 0.00, 0.00, 297
398, 7.11, 0.61, 43.53, 40.00, 185, 14.85, 0.00, 0.00, 36.00, 0, 0.00, 0.00, 0.00, 0.00, 302
55, 4.79, 2.40, 9.92, 2.00, 217, 3.83, 0.00, 0.00, 0.00, 0, 0.00, 0.00, 0.00, 0.00, 225
660, 0.00, 5.21, 67.85, 64.00, 165, 0.00, 0.00, 11.02, 13.00, 0, 0.00, 0.00, 0.00, 0.00, 501
93, 21.27, 18.80, 57.40, 38.00, 217, 0.00, 3.90, 13.78, 5.00, 0, 0.00, 0.00, 0.00, 0.00, 223

660, 2.62, 5.13, 25.60, 24.00, 74, 5.35, 0.00, 2.40, 1.00, 0, 0.00, 0.00, 0.00, 0.00, 425
398, 5.40, 3.95, 82.60, 58.00, 190, 0.00, 1.27, 6.22, 6.00, 0, 0.00, 0.00, 0.00, 0.00, 301

2.6 SEAS TRACE OUTPUT FILES

SEAS also provides an optional user trace of a simulation. The contents of the trace will vary, depending on which of the messages the user has chosen to include, but the output typically resembles the following:

151 BlueForce Unit_Sensor NightVision Sighting_dist: 4.706521 0.000000
151 BlueForce Weapon M16 Out_of_range 4.706521 0.393939
151 BlueForce Weapon M60 Out_of_range 4.706521 0.909091
151 BlueForce Weapon M242_Bushmaster Out_of_range 24.888192 1.818182
151 BlueForce Weapon M60 Out_of_range 24.888192 0.909091
151 BlueForce Weapon TOW-2 Out_of_range 24.888192 3.030303
151 BlueForce Weapon M242_Bushmaster Out_of_range 24.843965 1.818182
151 BlueForce Weapon M242_Bushmaster Target_in_range 0.774477 1.818182
151 RedForce Unit_under_fire Republican_Guard Fire_Strength/Vulnerability: 4.000000 4.000000
151 RedForce Unit_Body_Damage Republican_Guard KIA_At-Lon/Lat: 41.719026 32.244352
151 BlueForce Unit_Sensor NightVision Sighting_dist: 1.996864 0.000000
151 BlueForce Weapon M242_Bushmaster Out_of_range 1.996864 1.818182
152 BlueForce Unit_Sensor NightVision Sighting_dist: 3.797333 0.000000
152 BlueForce Comm: PRC25 Issuing_Situation_Report 1.500000 112.100000
152 BlueForce Weapon M16 Out_of_range 3.797333 0.393939

155 BlueForce Weapon TOW-2 Out_of_range 42.801172 3.030303
155 BlueRockets Weapon Cruise_missile Target_in_range 366.327710 727.272727
155 RedForce Unit_Collateral_Damage Mortar_Section Fire_Strength/Vulnerability: 25.000000 3.000000
155 RedForce Unit_Commo_Collateral_Damage Mortar_Section Radio_Hit_At-Lon/Lat: 41.727387 32.248194
155 RedForce Unit_under_fire Tawakalna_Division Fire_Strength/Vulnerability: 25.000000 3.000000
155 BlueForce Comm_Process_SitRep: PRC25 RedForce 17.185185 0.000000
155 BlueForce Weapon TOW-2 Out_of_range 20.823446 3.030303
155 BlueForce Weapon M242_Bushmaster Out_of_range 33.305776 1.818182

164 RedForce Vehicle_Sensor NightVision Sighting_dist: 1.060571 0.000000
164 RedForce Weapon 4.2_Inch_Mortar Target_in_range 1.060571 1.515152
164 BlueForce Unit_under_fire ScoutPlatoon Fire_Strength/Vulnerability: 4.000000 3.000000
164 BlueForce Vehicle_Weapon_Damage M3A2_Bradley Weapon_Hit_At-Lon/Lat: 41.822782 32.038046
164 RedForce Vehicle_Sensor NightVision Sighting_dist: 0.899645 0.000000
164 RedForce Weapon 4.2_Inch_Mortar Target_in_range 0.899645 1.515152
164 BlueForce Unit_under_fire ScoutPlatoon Fire_Strength/Vulnerability: 4.000000 3.000000
164 RedForce Sat Moly Lon/Lat: 160.500000 47.532001
165 BlueForce Unit_Sensor NightVision Sighting_dist: 1.297686 0.000000

179 RedForce Vehicle_Sensor NightVision Sighting_dist: 0.474300 0.000000
179 RedForce Weapon 4.2_Inch_Mortar Target_in_range 0.474300 1.515152
179 BlueForce Unit_under_fire ScoutPlatoon Fire_Strength/Vulnerability: 4.000000 3.000000
179 BlueForce Unit_Weapon_Damage ScoutPlatoon Weapon_Hit_At-Lon/Lat: 41.728857 32.245392
179 RedForce Vehicle_Sensor NightVision Sighting_dist: 1.416779 0.000000
179 RedForce Weapon 4.2_Inch_Mortar Target_in_range 1.416779 1.515152
179 BlueForce Unit_under_fire Cavalry_Troop Fire_Strength/Vulnerability: 4.000000 3.000000
179 BlueForce Unit_Commo_Damage Cavalry_Troop Radio_Hit_At-Lon/Lat: 41.737205 32.233481
179 RedForce Vehicle_Sensor NightVision Sighting_dist: 2.323830 0.000000
179 RedForce Weapon 4.2_Inch_Mortar Out_of_range 2.323830 1.515152

Gaps in the sequence numbers are places where several thousand messages have been removed. The trace output is voluminous, so the user probably won't want to print it, or trace on a regular basis. Values in the trace lines are either longitude / latitude, or distances and offensive / defensive values. Where questionable, it is labeled in the text.

3.0 SATISFYING THE SPECIFIED MINIMUM SEAS REQUIREMENTS

During the SEAS project, the United States Air Force defined a series of Specified Minimum Requirements to be met by the SEAS software at the end of SEAS Phase II activities. The following appendix identifies these specified requirements and describes the ways in which the requirements have been met.

3.1 SPECIFIED MINIMUM SEAS SYSTEM REQUIREMENTS

3.1.1 SENSORS

Represent coverage circles of sensors attached to satellites and aircraft platforms by projecting cone of specified half-angle from altitude of platform onto 2-D plane of battlefield. Register detection/location and identification of side, type object (e.g.., aircraft, TEL, truck, tank) and condition of object as alive or dead. User entered probability of detection (with specified location error), probability of identification and probability of correct condition assessment will contribute to stochastic behavior of simulation.

A cost attribute of each object is needed to compute cost of war. Sensors shall be logically sorted by two major attributes: designation of either active or passive sensor will allow representation of SIGINT behavior for detection of radar sensors and designation of environmental constraints on operation — daylight and clear weather, clear weather and insensitive to time of day, insensitive to weather or time of day.

Represent sensors attached to ground platforms (mobile or fixed) that project a sector of coverage in 2-D plane according to specified angular extent out to a specified range. Probability of detection/location, ident of side, type vehicle and condition shall also be determined. Terrain masking can be represented by adjusting range and probability of detection. Differentiation of sensor use against ground or airborne targets will allow the latter to detect aircraft by projecting sector coverage onto 2-D plane and testing against 2-D location of aircraft subpoint. Longer range given to SAM radars used against aircraft can compensate for greater coverage at flight altitude.

3.1.2 WEAPONS

Weapons attributes are minimum and maximum ranges, rounds available, resupply delay, CEP of aim point, Single Shot Prob Kill against type of target (read from lookup table). For targets in lookup table where SSPK is zero, weapon would never be employed against them. See rules under Units or Forces which determine how weapons allocated against targets. A cost attribute of each object is needed to compute cost of war.

3.1.3 COMMUNICATION SYSTEMS

Represent the communication network carrying sensor sitreps and commands according to the Order-of-Battle command hierarchy. Provide ability to add redundant nodes and links at user discretion. Control behavior of links according to range between nodes and vulnerability to jamming. Control behavior of nodes according to user specified time delay before message is relayed, reliability of relay, and also provide switch to control whether sit reps, commands or both should be relayed or not (CONOP issue).

A cost attribute of each object is needed to compute cost of war. Allow for permanent destruction of communication nodes when weapons successfully target the platforms to which communication devices are attached. Also allow for jamming by temporarily turning off receive and transmit capability of nodes within the range of given jammers. Allow specification of a communication device called a jammer which has a range over which it is effective. It will jam all communication nodes within its range, as long as they have an attribute vulnerable to jamming.

3.1.4 PLATFORMS

Represent airborne and ground platforms with altitude, average velocity, fuel consumption rate, fuel capacity, time to service, vulnerability cross section, capability flag (alive or dead). A cost attribute of each object is needed to compute cost of war and a manning attribute is needed to compute casualties. Aircraft are distinguished only by fact that altitude is not zero and they move at higher velocities. Platforms can have any combination of communication devices, sensors and weapons. When given a TAO either an aircraft platform or a ground platform will construct a regular orbit around boundaries of the TAO.

Platform rules take precedent over others. Such rules include simple ones for self-preservation such as stopping advance when under fire at a rate above a specified threshold (x) or those used to simulate the air campaign, such as move to assigned target location (taken from Target Priority list), shoot weapon/drop bomb and return to base (with slow velocity this behavior could be used to simulate SOF operations).

3.1.5 UNITS

Units can be composed of platforms or other units. They also have Rules of Engagement relative to when to attack, defend, avoid/retreat or surrender. They inherit the base of their parent Force although they are deployed spatially according alternative rules which can be selected by the user. Unit trigger levels for shifting from attack to defend, avoid or retreat can be set to estimate of force ratio or relative information advantage (estimated by sensor to shooter time delay on ones own side relative to that of enemy, i.e. ops tempo advantage).

3.1.6 FORCES

At the Force level, users can designate units which remain behind to secure the deployment base of operations and their form of defense (e.g., perimeter or fractile). One can also select movement rules from a list (e.g. in column or frontal advance and distance between objects as it

affects vulnerability to hostile fire). Rules govern allocation of weapons with ammo supply to fire at targets according to priorities entered by the user (eg., TELs, tanks artillery, APC).

3.1.7 SIDES

This definition allows forces of differing deployment bases to cooperate against a common enemy. Different Forces on the same side may be regarded as different services or Corps-level entities as well as different national units. One can construct a Force to represent non-combatant civilians and give it another side which is not allied with either warring side. The objects of this third side constitute a set of false targets which the ISR and command of either Red or Blue must discriminate in order to avoid wasting resources or causing negative political repercussions.

3.1.8 PLANS

Should allow three phases both Blue and Red and triggers for each phase based on perceived relative strength or relative information advantage. First phase might be air strikes but no ground activity or could also be ground attack. Second phase could have force switch to defensive mode from attack or vice versa. Third phase might be pursuit of retreating enemy.

3.2 IMPLEMENTED SEAS SENSOR OPERATIONS CAPABILITIES

3.2.1 IN GENERAL

- Sensors are described to SEAS by using the SEAS Deployment Editor.
- Each sensor has a unique name, and may have a detailed description.
- Each sensor may have unique characteristics, including:

  Minimum and maximum range
  Half angle of regard degrees
  Segment of energy spectrum
  Use limit
  Time limitations
  Other limitations
  Icon
  Location uncertainty
  Emission / Detection information

- Sensors may be owned by Units, Platforms, and Satellites.

3.2.2 AT INITIALIZATION TIME

- Sensors are created in general conformation with the user's description in the SEAS Deployment Editor.

- For sensors attached to sensors attached to Units and Ground Platforms, the maximum range is as specified by the user.

- For sensors attached to UAVs and aircraft, the maximum range is limited to the horizon at altitude. In the case of UAVs, this is a constant. For satellites the value is recalculated once each time step.

3.2.3 FOR EACH NEW SIMULATION ITERATION

- Any variables given a range by the user will be set to new values.

- Maximum range values are again limited as described in the section above.

3.2.4 SENSOR ACTIONS ON DEMAND

- Various internal commands are used to update location, altitude, etc. No user commands are available for sensor devices.

3.2.5 SENSOR ACTIONS AT EACH TIME INTERVAL

- Once each simulation, if the sensor is one that causes detectable emissions, it will register itself with the force for possible later detection.

- For satellites, a new horizon value is calculated based on current altitude AGL.

- For Sensors that "Detect" emissions, a check is made to find any emitting sensors within range. If found, they are reported as a threat.

- Once a non-detecting sensor knows where it is, and who or what it belongs to, the sensor will begin checking the location of each enemy unit, platform, plane. Each threat sensed will be reported as a threat.

- If the enemy threat is not "Alive," or hasn't been ordered to an initial location, no processing occurs.

- If the sensor use count has been exceeded, no further processing occurs.

- If the threat is out of range, no processing occurs.

- Once it has been determined that the unit is within the sensor's range, a check is made for sensing spectrum.

- For sensors in the spectrum "Touch," distance to the unit must be zero.

- For light sensors, two additional test are made. If the sensor has been defined as "Daylight Only," and it is currently dark where the unit resides, then the sensing process will fail. If the sensor is not marked as "Daylight Only," but the unit is in the dark, reduction of fifty percent in the detection probability will be designated.

- If the sensor is impaired by distance, a percentage is developed by dividing the distance by the maximum range. Random tests falling below this percentage will fail.

  If the sensor is time-constrained, a test is made to assure that the current simulation time is within the limitations.

  Half of all remaining units will be eliminated based on chance. The remaining units will generate a threat report.

  (Note that in the special case of sensors attached to a satellite, the sensors may or may not be driven each time step based on the user specified value for "Detection Probability." This value refers to the overall probability that an otherwise detectable unit will be seen while it is nominally visible.)

- When a unit or platform has been "Sensed", a threat report is made to the owner of the sensing device. A log file entry will be made specifying which type of owner the sensor belongs to, and the location of the threat.

- Threat reports include such information as threat location, speed, face angle, unit strength, distance, unit superior, enemy force, and which sensor reported the threat.

- Platforms that have been sensed will have a threat strength of 1 and additional information denoting that the threat is a platform. Special processing will occur based on the type of platform. e.g. Platforms based on the ground are susceptible to different weapons than those based in the air or on the seas.

3.2.6 DRAWING ROUTINE

- When requested, sensors are capable of drawing their footprint on the surface of the earth. They will not do so if they are attached to a satellite (Handled in satellite code) or their parent is dead.

- If the force is visible, and the user has allowed the display of sensors, the sensor is able to draw itself on request. All sensor footprints are drawn as bright green circles.

3.3 IMPLEMENTED SEAS WEAPON OPERATION CAPABILITIES

3.3.1 IN GENERAL

- Weapons are described using the SEAS Deployment Editor. Each has a unique name, and attributes that may include the following:

  Range in meters
  Kill radius in meters
  Lethality
  Dependability
  Rate of fire

Use limit
Icon
Land / Sea / Air / Space target indicators

- Weapons can be owned by Units, Planes, Platforms, and Satellites.

3.3.2 AT INITIALIZATION TIME

- During program initialization and again at the initialization of each simulation iteration, each of the weapon variables is set to its base value. If the user has specified a range of values for any variable, then a random selection is made within the specified value range.

3.3.3 WHEN A WEAPON IS UTILIZED

- If the weapon is a missile, and is currently in flight, no further action takes place here.
- If the weapon has been marked as being "Dead," no action is taken.
- If the threat is a unit and the unit has been marked as being "Dead," or is a dead platform, no action is taken. Further testing is made to determine if the weapon can be used against the target (Land versus Air, etc.)
- Distance to the threat is calculated.
- If the target is out of range, a message to that affect is sent to the log file and no further processing occurs.
- The rate of fire is examined to determine if the weapon can fire. For fractional rates of fire, a counter is kept until it exceeds the value 1. When this occurs the weapon will fire and the counter will be reset. For integer values one and greater, a times used counter is kept. When the times used per timer tic exceeds the rate of fire, the weapon is dysfunctional until the next timer tic.
- The use count is incremented. If there is a use limit, and the use count exceeds the use limit, no processing will occur.
- A message is sent to the log file.
- If a fractional random value is determined to be higher than the dependability, then no further processing occurs.
- A check is made to assure that the Force to which the threat belongs is an enemy force. If not, no further processing occurs.
- If the weapon is a missile, the goal and target information are stored away for later use. If the force is visible, the missile will show a launch blossom at the current location, then exit.

- For non-missile weapons, a single white line is drawn from the location of the weapon to the location specified in the threat report.
- The Unit , Plane, or Platform specified in the threat is told to "Take a Hit:"
- All damage and deaths are recorded and presented in the appropriate MOE fields output for statistical analysis.

3.3.4 WEAPON ACTIONS ON COMMAND

- Various internal commands are used to update weapon location and status. These are not available via external channels.
- For non-missiles, there are no actions taken on timer intervals other than specified internal commands and requested actions such as shooting at an entity.
- Missiles will update the location of the target and move toward the last known position of the target.

3.3.5 WHEN ASKED TO DRAW

- For missiles only, the current location of the missile (In flight) will be drawn.

3.4 IMPLEMENTED SEAS COMMUNICATION CHANNELS CAPABILITIES

3.4.1 IN GENERAL

- SEAS maintains 256 separate and unique channels of communications, and 256 separate channels for commands. Soon after program initialization, when the address of these queues is available, each Named Communications Channel requests an Index into the available channels. The channels are assigned a name on a first come first served basis. In the case of a jammer, the program request the Index for the Channel being Jammed rather than an Index for its own name. If multiple definitions exist for the same Named Channel, they will be given the same Index. The Index is then used to access the individual Situation Report Queues, and Command Queues.

3.4.2 INITIALIZATION AND NEW ITERATION

- Initial values are set to the values obtained by the parser.

3.4.3 GET STRENGTH

- If Run Status is "Alive" the value of the communication channel is added to the Dollar Value, otherwise it is added to the Damage Value.

3.4.4 PROGRAM TERMINATION

- Situation report queue [Index] and Command queue [Index] are emptied. Free element queues are emptied. All storage is returned to system.

3.4.5 WHEN ISSUING A COMMAND

- First, a probability test is made. If the probability test fails, the command is abandoned.
- Storage is obtained from the free element queue, or the system.
- The command fields are filled, and the command is placed on the Command queue [Index]. An origin identifier is placed in the message to prevent interactions of messages as they are re-broadcast.

3.4.6 BUILD MESSAGE ROUTINE

- A log message is built with different constants depending on the type of threat being processed (Platform, Unit, etc.)

3.4.7 REPORTING A THREAT

- If the run status is alive, and the communication channel is not a jammer, a probability test is made. If the test is survived, the following happens:
  - √ First, the threat is compared to all threats on the Situation Report Queue [Index]. If another threat exist for the same entity, the existing threat is updated, and a message is sent to the log file.
  - √ If no existing threat is found for the entity, a message is sent to the log, storage is acquired, and the threat is placed on the Situation Report Queue [Index].

3.4.8 ON MESSAGE RE-BROADCAST

- If the run status is "Alive" and the communication channel is not a jammer, a probability test is made. If the test is survived, the following occurs:

√ A check is made to assure the threat entity is not already on this queue. If it is, an immediate exit is made.

√ A re-broadcasting message is sent to the log.

√ Storage is acquired, and the threat is added to the Situation Report Queue [Index].

3.4.9 WHEN ASKED TO ACT

- Situation Report Queue [Index] and Command Queue [Index] are processed. If the delay is greater than zero, and a threat or command was placed on the queue by this communication channel ("Object") then the delay count is decremented by one. For threats, the Threat circle radius is increased based on the speed of the threat.

- If the delay is zero or less, the Sender will remove the threat/command from the queue, and will issue a message to the log.

- Threat elements for a communication channel's force are ignored.

- For enemy forces, an ArcDistance is calculated to the Broadcast Location. If in range, and this is a jammer, the threat will be take off the queue and a message issued to the log.

- Otherwise, if in range, and if an enemy, a log message is issued, the threat is Shot At, and the message reissued. After this has occurred, the force is notified of an "Alert," and the next threat is processed.

- Note that message propagation explosions are held in check in three ways. First, a message re-broadcast counter is held in each message. A message will not be rebroadcast more than kMessageQueueEntries times. Second, if a threat is already on queue, the threat will be updated rather than added a second time. Finally, the origin of the message is checked. Named Communications Channels will not rebroadcast their own threat.

3.5 IMPLEMENTED SEAS PLATFORM OPERATIONS CAPABILITIES

3.5.1 IN GENERAL

- Platforms are described to SEAS using the deployment editor. While there are many similarities between Platforms and SEAS Airplanes, their behavior is sufficiently different to warrant an individual object type.

- SEAS platforms represent vehicles, unmanned reconnaissance planes (UAVs), and non-mobile communications installations. They may or may not be able to move, on the ground, or in the air, and may or may not contain warm bodies. A local church on the battlefield is described in SEAS as a platform.

- Each Platform type has a unique name.
- Platforms may own weapons, sensors, and communications channels.
- UAVs can be assigned a Tactical Area of Operations (TAO). It will fly a circuit around the points in the TAO.
- Individual attributes of Platforms include how many warm bodies are on board, how much fuel is carried, and how fast it is burned.
- In addition, information is supplied by the user describing the location of operation (Air, Ground, Sea), the altitude of the Platform Above Ground Level (AGL) in meters, and the speed of the platform if it moves.
- Platforms may have an icon assigned to represent them on the screen. If no icon is present, they are depicted as a dot in the applicable Force color.
- Each type of Platform is assigned a particular degree of "Hardness" against weapons.

3.5.2 AT INITIALIZATION AND BETWEEN SIMULATIONS

- All attributes of the Platform are reset to their initial values.
- Variables are varied. If a TAO exists, a racetrack of coordinates is developed.
- Onboard fuel status is set to an arbitrary value within the limits of the platform.
- Statistical counters and health are restored.

3.5.3 AT TERMINATION

- Any orders remaining on queue are released.
- Sensors, Communication Channels, and Weapons are returned to the system.
- Any free storage held on queue for reuse is returned to the system.

3.5.4 WHEN ASKED TO "REPORT" A THREAT

- If the "Run Status" of the Platform is not "Alive," nothing is done.
- Each Communication Channel is requested to broadcast a "Report" of the threat information. This may or may not occur based on other considerations. See discussion on communications channels for more information.

- As each Communication Channel is accessed, a "Rebroadcast Counter" is incremented in the threat record. This is intended to keep messages from flooding the communication channels.

- Finally, the Platform will attempt to "Shoot At" the threat (see discussion below).

3.5.5 WHEN ASKED TO REBROADCAST THREAT WARNINGS

- A check is made to assure the Platform is still alive. If not, it returns immediately without performing any actions.

- A check is made to assure the threat rebroadcast count is less than the constant kMaxCommChannels (256). If not, the message has been on every conceivable channel, and will not be rebroadcast.

- Finally, Each communication channel is offered a chance to rebroadcast the threat warning. This may or may not occur.

3.5.6 WHEN SHOOTING AT A THREAT

- This routine is invoked when ever a threat is sensed by an onboard sensor, or is reported over a communications channel. Not all threats are actually fired at.

- If the Platform is not "Alive" nothing is done.

- An indicator is set to advance to combat speed.

- Each weapon on board is polled to see if it has the range/capability to fire on the target. Individual weapons make their own firing decisions.

3.5.7 WHEN QUERIED ABOUT THEIR STRENGTH

- Statistical fields containing information about body counts, weapons, communication channels, and sensors remaining and lost are passed to the caller.

- If the Platform is "Alive," a dollar value is reported. If not, then a dollar cost of battle is reported.

- Each communication channel owned, Sensor owned, and Weapon owned is asked to augment the report.

3.5.8 WHEN TAKING "FRIENDLY FIRE"

- A check is made to determine if the Platform is "Immune" to the weapon. If so, no processing occurs.

- If the Platform is already dead, no processing occurs.
- A central counter is updated noting the last time step when a shot was fired.
- The distance of the Platform from the impact sight of the weapon is compared to the Kill Radius of the weapon. If the Platform is outside the Kill Radius, no further processing is done.
- A message is sent to the log file indicating that the Platform is under friendly fire.
- A probability calculation is made based on the hardness of the Platform and the Lethality of the weapon. If judged improbable, no further processing occurs.
- The individuals lost is set to individuals on board.
- The Force individuals lost is updated.
- Friendly fire losses are updated.
- A message is sent to the log file indicating the Platform was lost to friendly fire.
- The Platform's owner is notified that an Platform has been lost.
- Each weapon, and communication channel owned by the Platform is marked as being "Dead." Relevant counters are set for each.

3.5.9 WHEN SHOT AT BY AN ENEMY

- A test is made to see if the Platform is immune to the weapon. If immune, a message is sent to the log file, and control returns to the caller immediately.
- If the Platform status is not "Alive," nothing more is done.
- A central counter is updated noting the last timer tic when a shot was fired.
- The distance of the Platform from the impact sight of the weapon is compared to the Kill Radius of the weapon. If the Platform is outside the Kill Radius, no further processing is done.
- A message is sent to the log file indicating that the Platform is under fire.
- A probability calculation is made based on the hardness of the Platform and the Lethality of the weapon. If judged improbable, no further processing occurs.
- The individuals lost is set to individuals on board.
- The Force individuals lost is updated.
- A message is sent to the log file indicating the Platform was lost to enemy fire.
- The Platforms unit is notified that an Platform has been lost.

- Each weapon, and communication channel owned by the Platform is marked as being "Dead." Relevant counters are set for each.

3.5.10 WHEN ACTING WITHOUT SPECIFIC ORDERS

- Each minute of simulation time, each Platform is requested to perform what ever automatic actions it is in need of performing. These may include housekeeping functions, the continuation of previously started orders, or updating the Platform location and notifying each sensor, communication channel, and weapon owned of the new location.

- Logic peculiar to UAV Platforms is implemented here: If their are no standing orders on queue, and if the Platform has been deployed with its unit. And the Platform (UAV) has been given a TOA, an order is generated to move to the next coordinate on the TAO. The user controls the flight path by the order in which the TAO coordinates are specified.

- Specific orders are executed on subsequent "minutes" of the simulation.

3.5.11 WHEN ACTING ON SPECIFIC ORDERS

- When jammers are ordered on or off, each of the communications channels is notified.

- Various housekeeping routines are available to the programmer. These have to do with assigning superiors, etc. and are not available to the user.

- When ordered to deploy with their unit, the Platforms will assume a location and offset relative to their owner. Or to the first point of their TAO.

- When told to move to a goal, the Platform will move at speed to the goal and update its current location. Weapons, communications channels, and sensors are apprised of the new location.

- When a goal is reached, a message is sent to the log file.

- After reaching a goal, the Platform will most likely be told to initiate a search for some specific objective. This might be a unit, or platform. If no specific objective is named in the target list, then no processing occurs. Otherwise, each sensor is told to search for the specific objective. If the specific objective was named but not found, the target is marked as being 100% destroyed. This stops future sorties from being flown against the target until the entire target list has been exhausted, when all targets are returned to active status.

- If the specific objective was found, a new goal is set, and a new MoveTo command is executed to position the Platform over the objective.

- When the new goal is reached, a message is sent to the log file.

- No logic specific to "Attack" is implemented. Actions in an attack are no different that those in a normal movement command.
- When told to defend, a non-UAV platform will assume a position with relative offset to its owner.

3.6 IMPLEMENTED SEAS UNIT OPERATIONS CAPABILITIES

3.6.1 IN GENERAL

- Units are described using the SEAS Deployment Editor. Each has a unique name and may have a detailed description.
- Units may be owned by a Force, or another Unit. There are no restrictions that preclude a unit being owned by more than one force or unit.
- Units may own other units, Sensors, Weapons, Communication gear, and / or platforms.
- Each unit has an estimated speed of travel, a unit head count, and a presumed offensive capability.

3.6.2 AT INITIALIZATION OF A SIMULATION ITERATION

- Unit strength is restored to its base value, and other variables are set to their starting values. Any variables selected by the user to act as independent variables are set to a random value within the specified range. Unit speed is set to the speed of the fastest sub unit or platform if it exceeds the user specified speed.
- An offset and angle are calculated for each platform owned by the unit. Each platform and airplane is notified of its place in the world. If the unit speed is less than that of any owned platform, it will be updated to the speed of the fastest platform.

3.6.3 AT PROGRAM TERMINATION

- All commands, Units, Platforms, Planes, Communication gear, Weapons, and Sensors are deleted.

3.6.4 WHEN SHOT AT

- First, a note is made that the unit is under fire.
- A message may be sent to the log file describing the situation.

- A determination based on the "Hardness" of the unit and the "Lethality" of the weapon to see if the weapon will have any effect. If it is determined that the unit sustained damage, the value of the offensive weapon will be subtracted from the current unit strength. A message is sent to the log file describing the current unit disposition.

- If it is determined that the unit has no live individuals left, (this value includes individuals in owned platforms) the unit is marked as being dead. In this instance, a message is sent to the log file, and all weapons owned by the unit are also marked as having been destroyed.

3.6.5 WHEN REPORTING A THREAT

- If the unit is alive, each Named Communication Channel that the unit has access to will be requested to broadcast a threat alert. As this happens, the rebroadcast count of the threat is increased once for each channel.

- After reporting the threat on all available communication channels, an attempt is made to shoot at the threat.

3.6.6 WHEN SHOOTING AT SOMETHING

- A command is passed to each sub-unit and platform requesting that the threat be dealt with. This will continue until all sub-units and platforms have been exhausted. In either case, the unit then proceeds to take its own shot by requesting all weapons to evaluate their effectiveness.

3.6.7 WHEN RE-BROADCASTING A THREAT

- A unit may be requested by a communication device to re-broadcast a threat on all available channels. When this occurs, the threat is first checked to assure that it has not passed the re-broadcast limit, and that the unit is in fact still alive. When these tests have been made, each piece of communication gear is requested to broadcast the threat. This may or may not be accomplished based on processes described in more detail in the communication device description. Basically, a threat will not be rebroadcast on the same frequency as the original threat report. Other conditions may apply.

3.6.8 WHEN PROCESSING SPECIFIC COMMANDS OR TIMER INTERVALS

- A check is made to determine if "One Time" and "Once Per Simulation" logic should be exercised. Routines may be executed and switches set based on the current situation.

- At timer intervals, the Unit ripples through all sub-units, platforms, planes, sensors, weapons, and communication gear driving their respective actions for timer intervals.

- When a specific command is passed down to a unit, the order is placed on a queue in FIFO fashion. Orders are taken from the queue and placed in "LastOrder." Repeated entry into the Act portion of the logic will continue execution of a particular order until the logic for that order clears the order from "LastOrder" so as to make room for the next command on queue.

- A request is made to identify the current order. This may be a previous order that is in the process of execution but has yet to finish, or it may be the next available order on the queue if all previous commands have been completed. In the case of a previous order, the typical action would be to continue the requested action until completed.

- Two specific internal orders, the first an abort command and the second requesting a new simulation to start will clear the current order and all orders waiting on the queue. Abort commands are also passed to sub-units and platforms owned by the unit. The command to begin a new simulation is passed to all sub-units, platforms, sensors, weapons, and communication gear owned by the unit.

- There are various internal commands used for program control and housekeeping that will not be covered in this description. They perform tasks such as notifying a unit which force it belongs to. The other commands are external commands supplied by the user and passed down from the force level to the units and platforms owned by the force. These commands are explained in detail as follows:

  √ JAMMER ON/OFF: These commands are passed on to all sub units, platforms, planes, and communication devices. The command is then discarded.

  √ WAIT: The unit will check the specified timer event for completion. If the event is complete, the current command is discarded. Otherwise, the current event stays in control and processing continues until the next timer interval when the timer event will be checked again. No other commands (Except abort and start new simulation) will be executed until the timer event occurs.

  √ DEPLOY: A test is made to determine if the unit was given its own deployment command, or inherited the command from the force. If the command was issued directly to the unit, the location of the deployment is verified, and the unit is positioned accordingly. If the command was issued to the force at large, the unit positions are determined based on an offset from the force location. In either event, spacing and offset information is sent to all sub units, platforms, and planes, and the current location information is sent to all weapons, sensors, and communication devices. Finally, the current command is cleared to allow the next command to be processed.

  √ DEFEND: This command is similar to deploy, but rather than placing the unit at the location immediately, the location plus offset is set as a goal, and the unit is moved to the location over time. The Finish Move routine is used to complete the transaction.

√ FINISH MOVE: If the current goal and the unit's current location are equal a log entry is made telling the user that the unit has reached its goal. The current command is then cleared to permit the execution of subsequent commands. If the goal and location are not equal and the unit is alive, the difference in location and the face angle are calculated. If the unit speed has been specified as being greater than zero, some movement will occur. The exact movement may vary based on a number of conditions. If the unit has a superior, and that superior is now dead (owning unit has been destroyed) then the goal of the current unit is set to the superior unit's goal plus a small offset. If the distance to the goal is less than the distance traveled at the current speed in one timer interval, the current location is set to the goal. Otherwise, the current location is adjusted by the distance which could be traveled at the current speed.

√ MOVE TO: A move command is begun by updating the unit's goal. The difference in location, and direction to the goal are then calculated. In preparation for the move, sub-units are directed to move into a trailing position and place their platforms into a protective echelon alternating right or left. Any platforms belonging the main unit are also moved into a protective echelon. When the formation is complete, each sub-unit is notified of its new goal, and the spacing it should strive for during the movement. Control is then passed to the Finish Move routine.

√ ATTACK: If an attack command is issued to a unit that has a speed of zero, it will be transmitted to all sub units prior to any other actions taking place.

- A maximum range one kilometer is assumed as a default value.

- If there are no subunits involved, a branch is made over the sub unit logic described below.

- Range and azimuth information are calculated to the attack location. An imaginary semicircle is developed at either the distance to the attack location (away from the goal) or 3 effective maximum ranges away, which ever is greater. Points on the semicircle are assigned to each sub unit, and a move to order is given to each unit. The parent unit assumes one point at the farthest end of the semicircle as a personal goal and moves itself and its platforms to that position.

- When the initial (point on the semicircle) position has been reached, the unit will order its platforms to move to the final goal. It will then initiate a five time step pause while the platforms race ahead.

- After the brief pause, the unit will attempt to reach the goal, but will do so with a slowed speed. After considerations for terrain are made, the speed will be adjusted to 0.25 that for units without platforms (Command units) and 0.5 the normal speed for units that have platforms. This is to insure that the unit leaders do not precede their platforms into battle.

- On completion of the command, the command is changed into a Defend command so that owned sub units and platforms will assume a defensive posture and positioning around the attacked location.

3.6.9 WHEN CALCULATING CURRENT UNIT STRENGTH

- Units are periodically requested to report on their current strength levels. When this occurs, they in turn request the information from any sub-units and platforms that may be owned. The information is then passed onto the requester. This information becomes the basis for the various MOEs, and is used at the force level to make several decisions.

3.6.10 WHEN ASKED TO DRAW

- If Graphics have been turned off, no action occurs.
- Each sub-unit, platform, and communication device, airplane, and weapon owned by the unit is asked to draw itself.
- The current screen location is calculated.
- If the unit is alive and the force is visible the unit will either display the assigned icon, or draw a small dot in the force color.
- If the force is visible but the unit is dead, the unit will display a small white dot.

3.7 IMPLEMENTED SEAS FORCE OPERATIONS CAPABILITIES

3.7.1 IN GENERAL

- Forces are created using the SEAS Deployment Editor. Each force must have a unique name, and may contain a detailed description.
- Each force may have both enemies and allies.
- Each force may own one or more units.
- During the planning process, one or more satellites may be attached to a force.
- A force can be assigned a specific color, and an icon.
- Each force is assigned a specific *Battle Stance* that can be *Attack*, *Defend*, or *Avoid*. In addition to the basic stance, two force ratios are supplied by the user.
  - √ The first is n:1 permitting the unit to attack if it has an n:1 advantage and no other orders.
  - √ The second ratio specifies that if the stance is *Avoid* and the force has an n:1 disadvantage it may quit the field.

- Finally, when describing a force, the user must supply a percentage for withdrawal. If a force is assigned seventy percent force ratio for combat termination, for example, it will withdraw from a confrontation when thirty percent of its strength has been exhausted.

3.7.2 AT PROGRAM INITIALIZATION

- All attached units and satellites are apprised of the force they belong to.
- Units are assigned an angle and offset distance from the force location.

3.7.3 AT THE START OF A NEW SIMULATION ITERATION

- Force strength and other fields are set to their base values.
- Threat levels are set to zero, and the threat location is set to the closest enemy location.
- The order queue for the units in a force is emptied and set to contain a fresh set of the orders given to this force.
- If withdrawal percentage has been specified as an independent variable, a new random value is chosen from the specified range of values.

3.7.4 WHEN GIVEN AN ALERT

- A test is made to determine if the threat has been reported in the current timer tic. Sightings of airplanes and UAVs are not used to develop a threat map. For units and platforms the following logic applies. If a threat was found to have been reported in the current timer tic, the threat location and strength are updated in place. (Airplanes and UAVs are entered in the threat map, but given a strength of zero to nullify their effect.) If the threat has not been seen previously in this timer tic a new entry is made to the threat map.
- The threat map queue is a circular queue containing kNumThreats entries. kNumThreats is currently set at 500. If the current threat entry is 500, it is modified to point to threat 0, the front of the queue.

3.7.5 WHEN PROCESSING A TIMER INTERVAL

- A test is made to see if "One Time" processing has occurred. If not, the one time processing is done, and the switch set to bypass this processing next time through the logic.
- If the parameters are null (Happens once each time step), each sub-unit and satellite is notified of the interval change. If the location of the main threat has not

been filled in, the location of the closest enemy unit or platform is placed in the threat location.

- For all commands and timer tics, the threat map is recalculated. The weighted (by strength) mean and standard deviation of all threats reported in the past 30 timer intervals are used to derive the boundaries of the threat map. The threat location is set to the weighted mean longitude / latitude of the known threats.

- If all commands on the force order queue have been exhausted, the queue is reset to its initial condition.

3.7.6 WHEN PROCESSING SPECIFIC COMMANDS

- Actions here vary by command as follows:

  √ MOVE TO: A test is made to verify the existence of the specified location. If the location is valid, all owned units are ordered to move to the specified location.

  √ DEPLOY TO: The specified location is verified, and if good, the current location is saved as the previous location. Each owned unit and their sub units are then ordered to deploy to the location.

  √ ATTACK: If the location is valid, each owned unit is assigned to attack the location.

  √ DEFEND: Again, assuming a valid location, each unit is assigned to defend the location.

  √ WAIT UNTIL: The specified timer event is check for consistency. If the timer event is a known value, all owned units are told to wait for the event to occur.

  √ JAMMER OFF: The Named Communication Channel's Index value to be jammed is derived, and each unit notified to turn its jammers off.

  √ JAMMER ON: The Named Communication Channel's Index value to be jammed is derived, and each unit notified to turn its jammers on.

- When a new simulation begins, the Force notifies each unit of the requested spacing interval it is to assume. Each unit and satellite is notified that a new simulation has begun, and the threat map queue is cleared.

3.7.7 WHEN REQUESTED TO REPORT ON FORCE STRENGTH

- Each owned unit (and all of their sub-units) are queried as to their current strength levels.

- If the current force strength is less than that specified by the user for simulation termination, preparations are made to end the current simulation and begin another.

- If the current strength has been diminished to the user specified withdrawal level, all units are ordered to abandon their current orders. Also, if the force stance is "Attack," it will be changed to "Defend" in order to facilitate a withdrawal from enemy contact.

- Information is collected for all units needing orders. If they exist, a message is sent to the log file indicating that there have been units found that did not have standing orders. (This could also mean that the units have been ordered to abandon their current orders.)

- For each unit needing orders a determination is made based on the force stance. The logic is as follows:

- When the force stance is "Attack," a message is placed in the log file showing the current force strength and the perceived threat strength. The location of the closest enemy is determined. If the current force strength is greater than the user specified ratio to the perceived enemy strength, an order is given to the unit to attack the enemy.

- When the force stance is "Defend," a determination is made as to the location of the closest enemy unit or platform location. An initial order is prepared telling the unit to move one and one half times the maximum effective weapons range away from the enemy location. A message is sent to the log indicating this proposed move. However, if the current force strength has fallen below the withdrawal limit, and the unit is within two effective weapons ranges to the enemy, the order is modified to move three effective maximum weapon ranges away from the enemy. A message describing this modification is sent to the log file. In either case, a check is made to see if the unit has an enemy. If an enemy exists, and if that enemy has a stance of attack, and the current strength is such that the enemy would normally attack, and the enemy units do not have standing orders, the enemy is notified of the withdrawal and told to pursue the retreating unit.

- When the force stance is "Avoid," the location of the closest enemy unit or platform is found. A message is sent to the log showing the distance to the enemy, and the maximum effective range of the weapons assigned to the unit. The unit is ordered to move a safe distance away from the enemy, then ordered to return to its original base. Both of these commands are formatted as messages and sent to the log file. If an enemy exists, and if that enemy has a stance of attack, and the current strength is such that the enemy would normally attack, and the enemy units do not have standing orders, the enemy is notified of the withdrawal and told to pursue the retreating unit.

- A test is made to assure that the location given in the movement or attack command is within the range displayed on the current user display screen. If it is not, the unit is directed back toward the center of the users screen.

- Finally, the queue of units needing orders is deleted, and the strength of the force returned to the caller.

3.7.8 WHEN REQUESTED TO DRAW

- If the user has opted to view graphic output, and the force is visible, and the user has requested to see the threat map (All default values), then painting of a threat rectangle will proceed.
- The initial color of the threat rectangle is set to the Force colors. Next, if a list of enemies exists, the color of the force rectangle is set to the color of the first force on the enemy list. Then, if the threat strength is greater than zero, the threat rectangle is drawn.
- Next, each unit belonging to the force is requested to draw itself.
- Each satellite belonging to the force is asked to draw itself.

3.8 IMPLEMENTED SEAS AIRPLANE OPERATIONS CAPABILITIES

3.8.1 IN GENERAL

- Airplanes are described to SEAS using the deployment editor. While there are many similarities between Airplanes and SEAS Platforms, their behavior is sufficiently different to warrant an individual object type.
- Each Airplane type has a unique name.
- Airplanes may own weapons, sensors, and communications channels.
- Individual attributes of Airplanes include how many individuals are on board, how much fuel is carried, and how fast it is burned. In addition, information is supplied by the user describing the altitude of the Airplane Above Ground Level (AGL) in meters, and the speed of the aircraft.
- Airplanes may have an icon assigned to represent them on the screen. If no icon is present, they are depicted as a green dot.
- Each type of Airplane is assigned a particular degree of "Hardness" against weapons.

3.8.2 AT INITIALIZATION AND BETWEEN SIMULATIONS

- All attributes of the Airplane are reset to their initial values.
- Variables are varied.
- A delay for refueling is set at 0 to 30 minutes to stagger refueling operations between multiple airplanes.
- Statistical counters and health are restored.

3.8.3 AT TERMINATION

- Any orders remaining on queue are released.
- Sensors, Communication Channels, and Weapons are returned to the system.
- Any free storage held on queue for reuse is returned to the system.

3.8.4 WHEN ASKED TO "REPORT" A THREAT

- If the "Run Status" of the Airplane is not "Alive," nothing is done.
- Each Communication Channel is requested to broadcast a "Report" of the threat information. This may or may not occur based on other considerations. See discussion on communications channels for more information.
- As each Communication Channel is accessed, a "Rebroadcast Counter" is incremented in the threat record. This is intended to keep messages from flooding the communication channels.
- Finally, the Airplane will attempt to "Shoot At" the threat. See discussion below.

3.8.5 WHEN ASKED TO REBROADCAST THREAT WARNINGS

- A check is made to assure the Airplane is still alive. If not, it returns immediately without performing any actions.
- A check is made to assure the threat rebroadcast count is less than the constant kMaxCommChannels (256). If not, the message has been on every conceivable channel, and will not be rebroadcast.
- Finally, Each communication channel is offered a chance to rebroadcast the threat warning. This may or may not occur.

3.8.6 WHEN SHOOTING AT A THREAT

- This routine is invoked when ever a threat is sensed by an onboard sensor, or is reported over a communications channel. Not all threats are actually fired at.
- If the Airplane is en-route to target on a sortie, no action is taken against the threat.
- If the Airplane is not "Alive" nothing is done.
- An indicator is set, telling the pilot that (s)he should advance to combat speed.
- A distance to target is calculated. If it is greater than the maximum range of all weapons onboard, then nothing more is done.

- Each weapon on board is polled to see if it has the range / capability to fire on the target. Individual weapons make their own firing decisions.

3.8.7 WHEN QUERIED ABOUT THEIR STRENGTH

- Statistical fields containing information about body counts, weapons, communication channels, and sensors remaining and lost are passed to the caller.
- If the Airplane is "Alive," a dollar value is reported. If not, then a dollar cost of battle is reported.
- Each communication channel owned, Sensor owned, and Weapon owned is asked to augment the report.

3.8.8 WHEN TAKING "FRIENDLY FIRE"

- A check is made to determine if the Airplane is "Immune" to the weapon. If so, no processing occurs.
- If the Airplane is already dead, no processing occurs.
- A central counter is updated noting the last timer tic when a shot was fired.
- The distance of the Airplane from the impact sight of the weapon is compared to the Kill Radius of the weapon. If the Airplane is outside the Kill Radius, no further processing is done.
- A message is sent to the log file indicating that the Airplane is under friendly fire.
- A probability calculation is made based on the hardness of the Airplane and the Lethality of the weapon. If judged improbable, no further processing occurs.
- The bodies lost is set to bodies on board.
- The Force bodies lost is updated.
- Friendly fire losses are updated.
- A message is sent to the log file indicating the Airplane was lost to friendly fire.
- The Airplanes unit is notified that an Airplane has been lost.
- Each weapon, and communication channel owned by the Airplane is marked as being "Dead." Relevant counters are set for each.

3.8.9 WHEN SHOT AT BY AN ENEMY

- A test is made to see if the Aircraft is immune to the weapon. If immune, a message is sent to the log file, and control returns to the caller immediately.
- If the Aircraft status is not "Alive," nothing more is done.
- A central counter is updated noting the last timer tic when a shot was fired.
- The distance of the Airplane from the impact sight of the weapon is compared to the Kill Radius of the weapon. If the Airplane is outside the Kill Radius, no further processing is done.
- A message is sent to the log file indicating that the Airplane is under fire.
- A probability calculation is made based on the hardness of the Airplane and the Lethality of the weapon. If judged improbable, no further processing occurs.
- The individuals lost is set to the number of individuals on board.
- The Force individuals lost is updated.
- A message is sent to the log file indicating the Airplane was lost to enemy fire.
- The Airplanes unit is notified that an Airplane has been lost.
- Each weapon, and communication channel owned by the Airplane is marked as being "Dead." Relevant counters are set for each.

3.8.10 WHEN ACTING WITHOUT SPECIFIC ORDERS

- Each minute of simulation time, each Airplane is requested to perform what ever automatic actions it is in need of performing. These may include housekeeping functions, the continuation of previously started orders, or updating the Airplane location and notifying each sensor, communication channel, and weapon owned of the new location.
- Logic peculiar to Airplanes is implemented here: If their are no standing orders on queue, and if the Airplane has been deployed with its unit. Then if the unit has been given a Target List, and no active target is known, if the Airplane is refueling, the refueling continues. Otherwise, a search is made for the highest priority target that is still valid. When a target is found, a search is made to determine if the correct weapons are onboard to attack the target. If they are, orders are generated to move the Airplane to the target's initial point, to search for the specific objective of the target list entry, to attack the specific objective, and to return to base for refueling and rearming.
- If the weapons specified by the user for use against the specific objective of the target list entry are not onboard, then a second search is made to identify a target against which the available weapons stores can be used. If a target is found, orders

are generated as above. Otherwise, the highest priority target is accepted without regard to weapons load. Orders are generated as above.

- If no targets are available, the Airplane designates the center of the perceived threat map as the target area, and applicable orders are generated.
- Specific orders are executed on subsequent time steps of the simulation.

3.8.11 WHEN ACTING ON SPECIFIC ORDERS

- When jammers are ordered on or off, each of the communications channels is notified.
- Various housekeeping routines are available to the programmer. These have to do with assigning superiors, etc. and are not available to the user.
- When ordered to deploy with their unit, the airplanes will assume a location and offset relative to their owner.
- When told to move to a goal, the airplane will move at speed to the goal and update its current location. Weapons, communications channels, and sensors are apprised of the new location.
- When a goal is reached, a message is sent to the log file.
- After reaching a goal, the Airplane will most likely be told to initiate a search for some specific objective. This might be a unit, or platform. If no specific objective is named in the target list, then no processing occurs. Otherwise, each sensor is told to search for the specific objective. If the specific objective was named but not found, the target is marked as being 100% destroyed. This stops future sorties from being flown against the target until the entire target list has been exhausted, when all targets are returned to active status.
- If the specific objective was found, a new goal is set, and a new MoveTo command is executed to position the Airplane over the objective.
- When the new goal is reached, a message is sent to the log file.
- When the new goal is reached, an indicator is set to note that the Airplane is now on the return leg of the sortie, and is now permitted to accept targets of opportunity if the weapon load remaining permits.
- Each enemy unit, platform, and airplane within the Kill Radius of the weapons load is fired on to the extent possible with the existing weapon load. This may have been limited by the user's inclusion of a "Use Limit" on bombs, etc. If this is the case, the weapons will fire to the extent possible, and will be "Re-Armed" when the Airplane returns from its sortie.
- A BDA assessment is made and the status of the target is updated to reflect the perceived damage.
- The Airplane is moved back to its base.

- A refueling layover of 20 minutes is started.
- All weapons on board the Airplane are re-armed.
- Past target pointer is removed and process begins anew.

3.9 IMPLEMENTED SEAS SATELLITE OPERATIONS CAPABILITIES

3.9.1 IN GENERAL

- One or more satellites may be attached to any Force defined via the SEAS Deployment Editor.
- Satellites and their properties are defined using the SEAS Deployment Editor.
- Satellites may act as the parent of Sensors, Weapons, and Communication Gear.
- When defining a satellite, the user may specify an overall communications delay experienced by all communication gear attached to the satellite.
- Each satellite is assigned a "Detection Probability" that affects each sensor attached to the satellite each simulation interval.
- An icon can be specified to represent the portrayal of the satellite during the simulation.
- Each satellite must be assigned an "Orbital data" file describing its location and altitude for the duration of the simulation. These orbital data files are produced by first using the SOAP program to determine and log the satellite's orbit, then using the READSOAP program to convert the SOAP log into a usable format. Since these files tend to be large, the user may also specify an "Offset" in minutes for the orbital data. This way, multiple satellites can share one orbital data file, each flying at a particular offset to the first specified position.

3.9.2 AT INITIALIZATION TIME

- The orbital data file is found, or read into storage and stored in a matrix of compressed values. Values processed from the file include latitude, longitude, and Kilometers away from the Earth's center (Altitude). Based on this information, and the "Half Angle of Regard" specified for sensors, the distance to the Earth's horizon is calculated. All of the above information is stored for later processing.

3.9.3 FOR EACH NEW ITERATION (NEW SIMULATION)

- All measure of effectiveness values (MOEs) are reset to their baseline value. If the user has specified variations in the values for communication delays, or detection probability, the appropriate values are varied.

3.9.4 AT DESTRUCTION TIME

- Sensors, communication gear, and weapons owned by the satellite are deleted along with any icon that may have been specified. If a "Region" control block has been created describing the "Footprint" of the satellite, it is also deleted.

3.9.5 WHEN A THREAT IS REPORTED

- If communication gear exists, and the communications delay specified for the communication gear is less than the overall delay specified for the satellite, then the communication gear delay is set to the overall delay specified for the satellite.
- For each piece of communication gear owned by the satellite, the threat report is propagated. e.g. The threat will be "Re-broadcast" on all active frequencies.
- After running through all owned communication gear, an attempt is made to "Shoot At" the threat.

3.9.6 WHEN ATTEMPTING TO "SHOOT AT" A THREAT

- An attempt is first made to determine if we have "Shot" at the threat coordinates before this shot, but during the same timer interval. If the threat has been processed during the same interval, a return is made to the caller.
- Assuming we have not shot at the threat during this timer interval, a record is made of this attempt. These records are limited to ten locations (In the case of a satellite). If ten locations have been processed during the timer interval, no shot will be taken.
- If the test above permit a shot, the satellite will try each available weapon until one of the weapons gets off a good shot.

3.9.7 WHEN A RE-BROADCAST OF A THREAT IS REQUESTED (BY COMMUNICATION GEAR)

- If the threat has been re-broadcast too many times, no action is taken.

- Otherwise, the satellite will attempt to rebroadcast the threat on all available communication gear.

3.9.8 SATELLITE ACTIONS ON DEMAND

- A variety of commands may be sent to the satellite from the owning Force. These commands have to do with house-keeping, and new simulation processing.

3.9.9 SATELLITE ACTIONS EACH TIMER INTERVAL

- The "Current" interval counter is increased and used as an index into the orbital data matrix saved during program initialization. If the interval counter exceeds the number of records found in the orbital data matrix, the counter is reset to zero.
- Based on the current latitude, longitude, altitude, and distance to horizon, an N pointed polygon is constructed describing the satellite's footprint. These points are then converted into a "Region" that is subsequently used by sensors to help determine the visibility of targets.
- An entry is made into the log file stating the location of the satellite.
- The current location and altitude are forwarded to each Sensor, Communication device, and Weapon for use in their calculations.
- Finally, each Sensor, Communication device, and Weapon is driven with a timer interval notification.

3.9.10 DRAWING ROUTINE

- If graphics are on, and the footprint of the satellite is visible on the current map, the satellite will draw its footprint, and any ICON that may be associated with the satellite.

3.10 IMPLEMENTED SEAS SIDES CAPABILITIES

This definition allows forces of differing deployment bases to cooperate against a common enemy. Different Forces on the same side may be regarded as different services or Corps-level entities as well as different national units. One can construct a Force to represent non-combatant civilians and give it another side which is not allied with either warring side. The objects of this third side constitute a set of false targets which the ISR and command of either Red or Blue must discriminate in order to avoid wasting resources or causing negative political repercussions.

A third force can be created, unallied with any other force. However, no force will shoot at it, for any reason. It is not possible to define a "side" per se; instead, the user can define a group of forces that are allied together, and in opposition to another group of forces.

3.11 IMPLEMENTED SEAS PLANS CAPABILITIES

The SEAS system should allow three phases both Blue and Red and triggers for each phase based on perceived relative strength or relative information advantage. First phase might be air strikes but no ground activity or could also be ground attack. Second phase could have force switch to defensive mode from attack or vice versa. Third phase might be pursuit of retreating enemy. The user can select a rule of engagement that permits the pursuit of retreating forces as part of an overall user-specified plan.

4.0 THE SYNTAX OF TACLAN — THE TACTICAL LANGUAGE FOR SEAS

This section provides details of TacLan, the tactical language for the SEAS system.

4.1 LEXICAL SPECIFICATIONS

a. The TacLan alphabet is all ASCII printable characters.

b. Every line of TacLan is surrounded by double-quote marks.

c. TacLan tokens are alphameric strings and punctuation marks. The alphabetic characters are {a-z, A-Z, _ }.

d. Lexical primitive tokens are: **integer**, **float**, **name**, **string**, **DOSpath**.

e. TacLan strings are delimited like C comments: /* this is a string */

f. A **name** is an alphameric token that begins with a letter.

g. A **DOSpath** is a DOS file specification. It may be a full, partial, or relative path, or just a filename.

4.2 SYNTACTIC SPECIFICATIONS

a. Every transition symbol is written in *italics.*

b. Every lexical token is written in **bold**.

c. The BNF symbol <== means "is replaced by".

d. The BNF symbol | means “or”.

e. The BNF symbol * means “repeated 0 or more times”.

f. BNF curly braces surrounding a phrase indicates the phrase is optional.

g. A complete TacLan deployment is a *deployment.*

h. The grammar that follows is yacc-compatible.

4.3 SEMANTIC NOTES

a. Semantic notes are given after a double-slash (//).

b. Although English unit parameters were changed to metric, the database field names have not been changed, unfortunately.

4.4 BNF SYNTAX FOR TACLAN

The BNF syntax for TacLan follows:

deployment <== *plan_header plan_body* plan_footer*

plan_header <== **Begin_SEAS** (*version*) **_plan_for** :
plan_name iterations cutoff

plan_footer <== **End_SEAS_plan_for** : *plan_name*

version <== **float** // format is: v.yymmdd

plan_name <== **name** // the name of this deployment

iterations <== **integer** // the number of iterations to perform

cutoff <== **integer** // the overall criterion for end of battle

plan_body <== *map_1* | *map_2* | *area* | *location* | *target_list* |
event | *sensor* | *comm* | *satellite* | *weapon* |
vehicle | *unit* | *airplane* | *force*

map_1 <== **Map1 : DOSpath**

map_2 <== **Map2 : DOSpath**

area <== **Begin_TAO** : *area_name*
*point**
End_TAO : *area_name*

point <== **Point : float float** // latitude and longitude

area_name <== **name**

location <== **Begin_Location** : *loc_name*
description
Longitude : **float**
Latitude : **float**
End_Location : *loc_name*

loc_name <== **name** // name of a location

description <== **Description** : **string**

target_list <== **Begin_Target_List** : **name**
*target**
End_Target_List : **name**

target <== **Begin_Target** : *targ_name*
{ **Objective** : > **name** } // for information only
{ **Priority** : **integer** }
{ **Weapon** : *weap_name* }
{ **Service** : **name** }
{ **Encyclo** : **name** }
End_Target : *targ_name*

targ_name <== **name**

event <== **Begin_Timing_Event** : *event_name*
description
Start_Year : **integer**
Start_Month : **integer**
Start_Day : **integer**
Start_Minute : **integer**
End_Timing_Event : *event_name*

event_name <== **name**

sensor <== **Begin_Sensor** : *sens_name*
description
*sens_parm**
End_Sensor : *sens_name*

sens_name <== **name** // the name of a sensor

sens_parm <== **Min_Range** : *float_var* |
Max_Range : *float_var* |
Degrees_Of_Reference : *float_var* |
Impaired_By_Distance : *boolean* |
Impaired_By_Weak_Signal : *boolean* |
Daylight_Only : *boolean* |
Impaired_By_Climate : *boolean* |
Time_Constrained : *boolean* |
Start_Time : *time* |
End_Time : *time* |
use_limit |
Segment_Of_Spectrum : **integer** |
icon |
LUM : *float_var* | // location uncertainty in meters
ProbIFFErr : *float_var* | // probability of IFF error
value | // dollars
Active : *boolean* | // true for active sensors (radar)
Detects : *boolean* // true if detects active sensors

boolean <== **integer** // 0 for false, -1 for true

float_var <== **float** | **float float** // two values if experimental

int_var <== **integer** | **integer integer** // two values if experimental

time <== 0 | *hours minutes seconds* | *hours minutes seconds* **PM**

hours <== **integer**

minutes <== **integer**

seconds <== **integer**

icon <== **ICON** : **integer** // icon resource index

value <== **Value** : **float** // in dollars

use_limit <== **Use_Limit** : *int_var* // how many times it can be used

comm <== **Begin_Comm_Gear** : *com_name*
description
Owner : **name** // force that owns this comm
Max_Transmission_Distance : *float_var*
Communication_Delay_In_Minutes : *float_var*

ProbabilityOK : float // probability of good transmission

{ *jammer* }

{ *value* }

End_Comm_Gear : *com_name*

com_name <== **name** // the name of a type of comm gear

jammer <== **Jammer** : -1 **Jams name** | **Jammer** : 0

satellite <== **Begin_Satellite** : *sat_name*

description

Path_To_Orbit_Data : **DOSpath**

Offset_To_Orbit_Data : **integer**

Communications_Delay : *int_var* // in minutes

Probability : *float_var* // of detection, per minute

*equipment**

{ *icon* }

{ *value* }

End_Satellite : *sat_name*

sat_name <== **name** // the name of a satellite

equipment <== **Sensor** : *sens_name* |

Weapon : *weap_name* |

Comm_Gear : *com_name* |

Vehicle : *veh_name* | // allowed only in units

AirPlane : *plane_name* // allowed only in units

weapon <== **Begin_Weapon** : *weap_name*

description

*w_parm**

End_Weapon : *weap_name*

weap_name <== **name** // the name of a weapon

w_parm <== **Range_In_Meters** : *float_var* |

Kill_Radius : *float_var* | // in meters

Fatality_Rate : *float_var* |

use_limit |

Rate_of_Fire : *float_var* |
lethality |
hardness |
icon |
Land : *boolean* | // true if can hit objects on land
Sea : *boolean* | // true if can hit seaborne objects
Air : *boolean* | // true if can hit airborne objects
Space : *boolean* | // true if can hit objects in space
Radar : *boolean* | // true if can hit radar/radio
Missile : *boolean* | // true if this is a missile
SpeedKph : *int_var* | // speed if a missile
value

hardness <== **Hardness** : *float_var* // log-odds scale
lethality <== **Offensive_Power** : *float_var* // log-odds scale

vehicle <== **Begin_Vehicle** : *veh_name*
description
icon
hardness
Lbs_Fuel_Capacity : *float_var*
Speed_Min : *float_var* // now in kph
Speed_Max : *float_var* // now in kph
Speed_Cruise : *float_var* // now in kph
Fuel_Use_Min : *float_var* // lbs per minute at min speed
Fuel_Use_Max : *float_var* // lbs per minute at max speed
Fuel_Use_Cruise : *float_var* // lbs per min at cruising speed
*equipment**
{ **Land** : *boolean* } // true if land vehicle
{ **Sea** : *boolean* } // true if sea vehicle
{ **Air** : *boolean* } // true if airborne vehicle
{ **UAV** : *boolean* } // true if vehicle is a UAV
{ **TAO** : **name** } // tactical area of operation
{ *value* }

Altitude : *float_var* // in meters (if UAV)

Bodies : *int_var* // number of personnel

End_Vehicle : *veh_name*

veh_name <== **name**

airplane <== **Begin_AirPlane** : *plane_name*

description

icon

hardness

Lbs_Fuel_Capacity : *float_var*

Speed_Min : *float_var* // now in kph

Speed_Max : *float_var* // now in kph

Speed_Cruise : *float_var* // now in kph

Fuel_Use_Min : *float_var*

Fuel_Use_Max : *float_var*

Fuel_Use_Cruise : *float_var*

*equipment**

value

Altitude : *float_var* // now in meters

Bodies : *int_var* // number of personnel

End_AirPlane : *plane_name*

plane_name <== **name** // the name of an airplane

unit <== **Begin_Unit** : *unit_name*

description

icon

Speed : *float_var* // kph

HeadCount : *int_var*

hardness

*subunit** // subordinate units

*equipment**

{ **TAO** : **name** } // Tactical Area of Operation

{ **JIPTL** : **name** } // target list

{ **BaseLocation** : *loc_name* }

```
                    orders*
                    End_Unit : unit_name
unit_name     <==   name                    // name of a unit
subunit       <==   SubUnit : unit_name     // subordinate unit

orders        <==   DeployTo : loc_name |
                    MoveTo : loc_name |
                    Attack : loc_name |
                    Pursue : name |
                    Defend : loc_name |
                    JammerOn : name |
                    JammerOff : name |
                    WaitUntil : event_name

force         <==   Begin_Force : force_name
                    Color : integer         // RGB, packed into a decimal integer
                    icon
                    { stance }
                    AttackRatio : float_var // local force ratio for attack
                    Withdrawal : float_var  // casualty percent for withdrawal
                    WithdrawRatio : float_var // local force ratio for withdrawal
                    Interval : float_var    // distance between units
                    ally*
                    enemy*
                    unit*
                    recon_sat*
                    orders*
                    End_Force : force_name
force_name    <==   name
stance        <==   Attack | Defend | Avoid
ally          <==   Ally : force_name
enemy         <==   Enemy : force_name
unit          <==   Unit : unit_name
recon_sat     <==   Satellite : sat_name
```

5.0 BIBLIOGRAPHY

Dockery, J. T. and A. E. R. Woodcock, 1993. *The Military Landscape: Mathematical Models of Combat.* Abington, Cambridge: Woodhead Publishing Ltd.

Woodcock, A. E. R., L. Cobb, and J. T. Dockery, 1989. Models of combat with embedded C2. V: Cellular automata. *International C. I. S. Journal.* 3(3): 5-44.

☆U.S. GOVERNMENT PRINTING OFFICE: 1997-509-127-47127

MISSION OF ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Material Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.